ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

PHÁT HIỆN CỘNG ĐỒNG TRÊN ĐỒ THỊ VÀ ỨNG DỤNG TRONG PHÂN TÍCH MẠNG XÃ HỘI

HỘI ĐỒNG: KHOA HỌC MÁY TÍNH
GVHD: TS. Nguyễn An Khương
GVHD 2: TS. Lê Văn Tuấn
—o0o—
GVPB: TS. Trần Tuấn Anh
SVTH: Nguyễn Tấn Đức (1410950)

Thành phố Hồ Chí Minh, 5/6/2018

Lời cam đoan

Chúng tôi xin cam đoan rằng, luận văn tốt nghiệp "Phát hiện cộng đồng trong đồ thị và ứng dụng vào phân tích mạng xã hội" là công trình nghiên cứu của chúng tôi dưới sự hướng dẫn của TS. Nguyễn An Khương và TS. Lê Văn Tuấn, xuất phát từ nhu cầu thực tiễn và nguyện vọng tìm hiểu của bản thân chúng tôi. Ngoại trừ kết quả tham khảo từ các công trình khác đã ghi rõ trong luận văn, các nội dung trình bày trong luận văn này là kết quả nghiên cứu do chính chúng tôi thực hiện và kết quả của luận văn chưa từng công bố trước đây dưới bất kỳ hình thức nào.

Thành phố Hồ Chí Minh, 5/6/2018

Nhóm tác giả

Lời cảm ơn

Đầu tiên, chúng tôi xin gửi lời cảm ơn chân thành nhất đến TS. Nguyễn An Khương và TS. Lê Văn Tuấn, chính nhờ sự hướng dẫn tận tình cũng như những kiến thức khoa học mà các thầy truyền đạt đã giúp chúng tôi hoàn thành tốt nhất luận văn này.

Chúng tôi cũng gửi lời cảm ơn đến các giảng viên trường Đại học Bách Khoa Thành phố Hồ Chí Minh, đặc biệt là các thầy cô trong Khoa Khoa học và Kỹ thuật Máy tính, những người đã truyền đạt những kiến thức quý báu trong hơn bốn năm học qua.

Cuối cùng, chúng tôi xin gửi lời cảm ơn tới gia đình, bạn bè, những người đã động viên, hỗ trợ chúng tôi trong suốt thời gian hoàn thành chương trình bậc Đại học.

Thành phố Hồ Chí Minh, 5/6/2018

Nhóm tác giả

Tóm tắt luận văn

Phân tích mạng xã hội (*social network analysis*) là một vấn đề mang nhiều tính thách thức. Trong đó, việc phát hiện cộng đồng (*community detection*) là một nhiệm vụ quan trọng trong phân tích mạng xã hội. Đó là cách tìm các cá nhận trong một mạng xã hội có những đặc điểm giống hoặc tương đồng nhau. Có nhiều dạng cộng đồng khác nhau và các cồng đồng có thể xen kẽ, chồng lặp lên nhau. Trong luận văn, ta sẽ tìm hiểu các phương pháp để giải quyết vấn đề phát hiện cộng đồng trên đồ thị lớn và cách xử lý vấn đề phát hiện cộng đồng của mạng xã hội trên dữ liệu lớn hiện nay.

Với kết quả đạt được, chúng tôi hy vọng có thể đóng góp một phần vào các nghiên cứu trong lĩnh vực phân tích mạng xã hội sau này.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Chương 1

Tổng quan

1.1 Giới thiệu đề tài

Khái niệm mạng xã hội (*social network*) đã trở nên phổ biến sau sự ra đời của các mạng xã hội lớn như Facebook, Twitter, Zalo,... Có hai thuộc tính chính trong một mạng xã hội là các thực thể (*entities*) và các mối quan hệ (*relationships*) giữa các thực thể trong mạng. Các thực thể thường là người dùng và các mối quan hệ thường quan hệ bạn bè. Ngoài ra các thực thể cũng có thể là các trang web, các tổ chức và các mối quan hệ có thể là hợp tác, kinh doanh. Trong mạng xã hội, các mối quan hệ có thể ở dạng *có* hoặc *không* như Facebook hoặc nhiều mức khác nhau như Google+.
Xã hội học (*sociology*) là nghành khoa học nghiên cứ về xã hội đã bắt đầu và phổ biến từ nhiều thế kỷ trước nhưng cùng với sự phát triển mạnh mẽ của mạng máy tính (*Internet*) và các ứng dụng mạng xã hội đã tạo ra mối kết mọi trên toàn cầu và tạo ra một khối lượng dữ liệu (*data*) khổng lồ của thế giới thực để phục vụ cho nghiên cứu và phân tích. Dữ liệu này rất lớn, có thể đến hàng tỷ thực thể và các mối quan hệ nên ta cần tìm ra các phương pháp phù hợp để phân tích và xử lý trên mạng lưới khổng này.
Trên mạng xã hội thực tế, phân bố bậc của các thực thể (bậc: trên Facebook là số lượng bạn bè của một người) là không đồng đều, các mối quan hệ có thể dày đặc ở một số nhóm và thưa ở một số nhóm. Đặc tính số lượng cạnh nhiều hơn ở một số ở mạng xã hội thực được gọi là cấu trúc cộng đồng hay cấu trúc gom nhóm. Trong đề tài này, ta sẽ tập trung tìm hiểu các vấn đề xác định cộng đồ trên mạng xã hội và các phương pháp phát hiện công đồng trong mạng xã hội thực tế. Xem [1]

1.2 Mục tiêu và phạm vi đề tài

Các doanh nghiệp về truyền thông, Internet luôn muốn phân tích để khai thác các mối quan hệ và sự thay đổi của các mối quan hệ trên quy mô lớn trong một mạng xã hội đặc biệt là việc phát hiện cộng đồng, gom cụm người dùng để phục vụ cho các công việc như phân tích nhu cầu người dùng, marketing, phân bổ cơ sở hạ tầng,...
Có nhiều phương pháp để thực hiện việc gom cụm người dùng trên mạng xã hội như sử dụng vị trí (location), các mối quan hệ bạn bè, các nhóm cùng tham gia hay các đặc trưng khác như giới

tính, thời gian truy cập,... Tuy nhiên, đối với các mạng xã hội không có được các thông tin trên nên việc gom cụm người dùng qua các đặc trưng trên không thể thực hiện được. Trong luận văn ta sẽ tìm hiểu cách sử dụng mô hình đồ thị để mô hình hóa mạng xã hội và các giải thuật như Louvain, BigCLAM để tìm phát hiện cộng đồng trên đồ thị.

1.2.1 Mục tiêu của đề tài

- Tìm hiểu lý thuyết đồ thị, phương pháp mô hình hóa mạng xã hội về dạng đồ thị và các phương pháp giải bài toán phát hiện cộng đồng trên đồ thị.
- Đề xuất mô hình phù hợp để giải quyết bài toán phát hiện cộng đồng trong đồ thị lớn.
- Hiện thực các giải thuật phát hiện cộng đồng trên dữ liệu mạng xã hội thực.

1.2.2 Đối tượng, phạm vi và phương pháp nghiên cứu

Đối tượng nghiên cứu trong đề tài bao gồm:

- Các tài liệu, sách, báo liên quan tới lý thuyết đồ thị, lý thuyết mạng xã hội.
- Các phương pháp phát hiện cộng đồng trên đồ thị như Min-cut, BigCLAM.
- Công cụ hỗ trợ việc hiển thị đồ thị lớn giúp dễ dàng hình dung kết quả sau khi phát hiện cộng đồng.
- Sử dụng ngôn ngữ lập trình C++ và Python để xử lý dữ liệu mạng xã hội và hiện thực các thuật toán phát hiện cộng đồng trên đồ thị lớn.

1.3 Cấu trúc luận văn

Luận văn được chia làm sáu chương. Bố cục như sau: **Chương 1:** Đưa ra cái nhìn tổng quan về đồ thị mạng xã hội và phát hiện cộng đồng trên mạng xã hội.

Chương 2, Chương 3: Nêu ra những công trình liên quan về phát hiện cộng đồng và kiến thức nền tảng cho việc xây dựng mô hình. Nội dung hai chương này chủ yếu cung cấp cái nhìn tổng quan về vấn đề luận văn cần giải quyết cũng như kiến thức nền cho việc thực hiện các chương sau.

Chương 4, Chương 5: Đây là phần nội dung trọng tâm của luận văn. Hai chương này lần lượt trình bày phương pháp đề xuất và quá trình hiện thực mô hình.

Chương 6: Chương này trình bề về bộ dữ liệu, các phương pháp đánh giá kết quả và kết quả của quá trình thực nghiệm.

Chương 7: Tổng kết lại toàn bộ quá trình thực hiện, kết quả đạt được, những hạn chế và hướng mở rộng trong tương lai.

Chương 2

Mạng xã hội và đồ thị

2.1 Các loại mạng xã hội

Có nhiều loại mạng xã hội khác nhau và mỗi loại mạng xã hội có mỗi tính chất, đặc điểm khác nhau. Nhu cầu phân tích mạng xã hội ở các loại mạng này cũng khác nhau nên ta cần chia và phân tích tính chất của từng mạng đó. Nội dung mục này xem tại [1, 2]

2.1.1 Mạng xã hội bạn bè

Mạng xã hội có các đỉnh là con người và các cạnh là quan hệ của họ. Mạng xã hội này mục đích chính là kết bạn, nhắn tin và đăng các vấn đề để cùng nhau trao đổi ở mức độ bạn bè. Ta thường sử dụng đồ thị vô hướng không trọng sô cho loại mạng này. Tuy nhiên, nếu có nhiều mức độ bạn bè của họ thì ta có thể sử dụng trọng số để miêu tả. Ví dụ như Facebook, Weibo.

2.1.2 Mạng thư điện tử

Các đỉnh trong mạng biểu điễn cho một địa chỉ Email và cạnh thể hiện sự trao đổi thư qua lại lẫn nhau. Ta có thể sử dụng đồ thị có hướng có trọng số để miêu tả cho loại mạng xã hội này với hướng chỉ hướng gửi mail và trọng số chỉ lưu lượng mail được gửi đi. Phân tích mạng này có thể góp phần phát hiện ra các đối tượng gửi thư rác và phân loại email. Ví dụ như Gmail, Outlook.

2.1.3 Mạng công việc

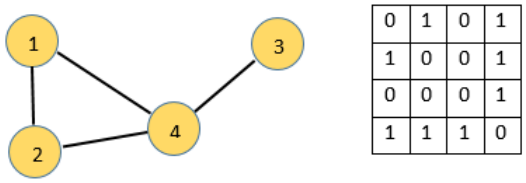
Mạng này dùng để lưu thông tin cá nhân, mối quan hệ việc làm tới các công ty và các quan hệ làm việc cùng nhau như cùng nhau đứng tên trong các bài báo khoa học hoặc cùng làm việc trong các dự án công nghiệp, kinh tế. Mỗi đỉnh là một người dùng và mỗi cạnh chỉ sự hợp tác giữa các đỉnh với nhau. Ví dụ như Linked-in.

2.2 Mô hình hóa một mạng xã hội

Đồ thị được xem là kiểu cấu trúc dữ liệu tốt nhất để mô hình hóa mạng xã hội. Đồ thị mạng xã hội giúp mọi người định nghĩa và hiển thị trực quan các thực thể cũng như mối quan hệ giữa các

thực thể có trong mạng. Ta định nghĩa một đồ thị $G = (V, E)$ với V là tập các đỉnh (*vertex*) hay gọi là các nút (*node*) và E là tập các cạnh (*edge*). Nếu tồn tại cạnh giữa 2 đỉnh u, v thì ta nói rằng u và v có mỗi quan hệ với nhau và cạnh giữa chúng được ký hiệu là (u, v). Ta mô hình hóa một mạng xã hội dưới dạng đồ thị bằng cách gán các thực thể cho các đỉnh và mối quan hệ giữa 2 thực thể cho các cạnh.
Đồ thị vô hướng thường được sử dụng để mô hình hóa mạng xã hội với vì nó phù hợp với bản chất mạng xã hội là các mối quan hệ qua lại lẫn nhau chứ không chỉ một chiều. Ngoài ra các đỉnh không có mối quan hệ với chính nó nghĩa là $(u, v) = 0$ cũng như giữa 2 đỉnh chỉ có duy nhất 1 cạnh.

Có nhiều cấu trúc dữ liệu có thể dùng để biểu diễn một đồ thị. Trong đó, cách đơn giản nhất là dùng một ma trận kề (*adjacency matrix*) A là một ma trận vuông kích thước VxV. Phần tử hàng i, cột j của ma trận A là thể hiện mối quan hện của thực thể thứ i và j trong mạng. Nếu thực thể thứ i và j có mỗi quan hệ với nhau thì $a_{ij} \neq 0$ và $a_{ij} = 0$ nếu ngược lại. Với đồ thị không trọng số thì $a_{ij} \in \{0, 1\}$. Với đồ thị có trọng số thì ma trận A chứa các giá trị của trọng số. Các phần tử trên đường chéo chính sẽ bằng 0. Với cách biểu diễn này, ta dễ dàng truy vấn được có tồn tại một cạnh giữa hai đỉnh i và j bất kỳ nào không. Tuy nhiên, để liệt kê tất cả các cạnh nối tới một đỉnh, ta cần duyệt tất cả các đỉnh trên đồ thị và điều này sẽ lãng phí thời gian nếu đồ thị ta đang xét là một đồ thị thưa.
Hình 2.1 mô tả một ma trận kể biểu diễn của một đồ thị đơn giản.

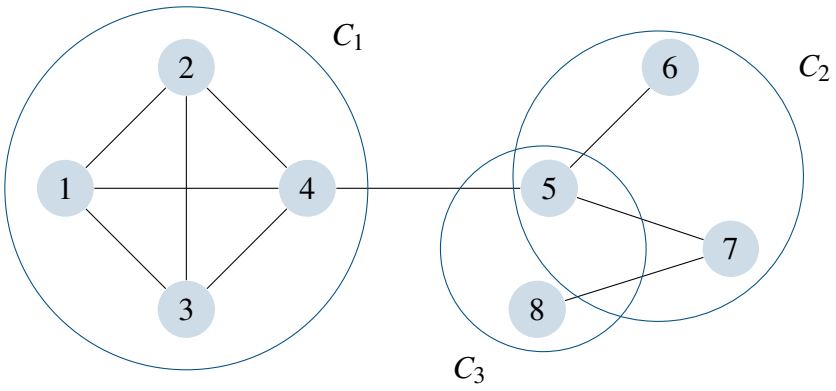


Hình 2.1. Đồ thị đơn (trái) và ma trận kề tương ứng (phải).

2.3 Cộng đồng và nhóm trong một đồ thị

Cộng đồng là một cấu trúc quan trọng trong đồ thị mạng, nơi các cá nhân hay các đỉnh trong cùng một cộng đồng được kết nối mật thiết với nhau hơn là với các cá nhân ở cộng đồng khác. Các cá nhân kết nói với nhau vì có sự quen biết hoặc có cùng thuộc tính như nhau. Vì vậy, ta có thể nhận xét rằng, các cá nhân cùng nằm chung một cộng đồng sẽ có những thuộc tính giống nhau.

Cho một đồ thị con (subgraph) C của đồ thị G với $|C| = n_C$ và $|G| = n$ đỉnh. Ta định nghĩa bậc nội bộ và bậc ngoài của đỉnh $v \in C$ là k_v^{int} và k_v^{ext} lần lược là số lượng cạnh kết nối từ v tới các đỉnh trong C và số cạnh kết nối từ v đến các đỉnh còn lại trong G. Nếu $k_v^{ext} = 0$ nghĩa là v chỉ có lân cận trong C, ta có thể nói v đang ở trong một cụm tốt. Nếu $k_v^{int} = 0$ nghĩa là v không có



Hình 2.2. Một đồ thị vô hướng

lân cận nào trong C, ta có thể nói v đang không nằm trong một cụm tốt và nó nên ở một cụm khác. Bậc nội bộ k_{int}^C của C là tổng các bậc nội bộ của các đỉnh của nó. Tương tự, bậc ngoài k_{ext}^C của C là tổng các bậc ngoài của các đỉnh của nó. Tổng bậc k^C là tổng của bậc của các đỉnh trong C. Nó xác định bằng công thức $k^C = k_{int}^C + k_{ext}^C$. Đồng thời, ta định nghĩa E_{C-C} là số cạnh nội bộ của đồ thị con C, E_C là tập cạnh có nối với các đỉnh thuộc C và tập E_C
E_{C-C} là tập các cạnh liên cụm.

Ta định nghĩa *mật độ cụm (intra-cluster density)* $\delta_{int}(C)$ của đồ thị con C là tỉ lệ giữa số lượng cạnh trong C trên số lượng tất cả các cạnh có thể

$$\delta_{int}(C) = \frac{|E_{C-C}|}{n_C(n_C-1)/2}. \tag{2.1}$$

Ta định nghĩa mật độ liên cụm (inter-cluser density) $\delta_{ext}(C)$ là tỉ lệ giữa số lượng cạnh nối từ C tới phần còn lại của G trên số lượng cạnh liên cụm tối đa có thể.

$$\delta_{ext}(C) = \frac{|E_C| - |E_{C-C}|}{n_C(n-n_C)}. \tag{2.2}$$

Đồng thời ta định nghĩa mật độ cạnh trung bình $\delta(G)$ của đồ thị G là tỉ lệ của số cạnh của G trên số cạnh tối đa có thể có $n(n-1)/2$.

$$\delta(G) = \frac{|E|}{n(n-1)/2}. \tag{2.3}$$

Với C là một cộng đồng, ta kỳ vọng rằng $\delta_{int}(C)$ lớn hơn mật độ liên kết trung bình $\delta(G)$ của đồ thị G. Đồng thời, $\delta_{ext}(C)$ nhỏ hơn so với $\delta(G)$. Việc tìm kiếm sự cân bằng giữa việc tăng $\delta_{int}(C)$ và giảm $\delta_{ext}(C)$ là mục tiêu của hầu hết các thuật toán phân cụm đồ thị. Một cách đơn giản, ta tìm cực đại của tổng hiệu số $\delta_{int}(C) - \delta_{ext}(C)$ trên tất cả các cụm.

Ví dụ 2.3.1. Cho đồ thị $G = (V,E)$ như Hình 2.2 có tập các đỉnh $V = \{1,2,3,4,5,6,7,8\}$ và tập các cạnh $E = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\},\{4,5\},\{5,6\},\{5,7\},\{7,8\}\}$. Đồng thời, ta xét 3 đồ thị con C_1, C_2, C_3 như trong hình. Ta tính được lần lược các mật độ trung bình của đồ thị, mật độ cụm và mật độ liên cụm như sau:

- Mật độ cạnh trung bình của G là $\delta(G) = \frac{|E|}{n(n-1)/2} = \frac{10}{8(8-1)/2} = \frac{10}{28} \approx 0.357$.
- Mật độ cụm của C_1 là $\delta_{int}(C_1) = \frac{|E_{C_1-C_1}|}{n_{C_1}(n_{C_1}-1)/2} = \frac{6}{4(4-1)/2} = 1$.
- Mật độ cụm của C_2 là $\delta_{int}(C_2) = \frac{|E_{C_2-C_2}|}{n_{C_2}(n_{C_2}-1)/2} = \frac{2}{3(3-1)/2} = \frac{2}{3} \approx 0.667$.
- Mật độ cụm của C_3 là $\delta_{int}(C_3) = \frac{|E_{C_3-C_3}|}{n_{C_3}(n_{C_3}-1)/2} = \frac{0}{2(2-1)/2} = 0$.
- Mật độ liên cụm của C_1 là $\delta_{ext}(C_1) = \frac{|E_{C_1}|-|E_{C_1-C_1}|}{n_{C_1}(n-n_{C_1})} = \frac{7-6}{4(8-4)} = \frac{1}{16} \approx 0.063$.
- Mật độ liên cụm của C_2 là $\delta_{ext}(C_2) = \frac{|E_{C_2}|-|E_{C_2-C_2}|}{n_{C_2}(n-n_{C_2})} = \frac{4-2}{3(8-3)} = \frac{2}{15} \approx 0.133$.
- Mật độ liên cụm của C_3 là $\delta_{ext}(C_3) = \frac{|E_{C_3}|-|E_{C_3-C_3}|}{n_{C_3}(n-n_{C_3})} = \frac{4-0}{2(8-2)} = \frac{1}{3} \approx 0.333$.

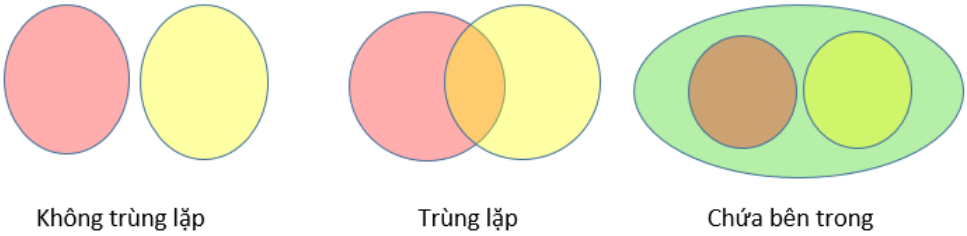
Dựa vào các giá trị vừa tìm được, ta có nhận xét như sau:

- Với cụm C_1, ta có mật độ cụm $\delta_{int}(C_1) = 1 > \delta(G)$ và mật độ liên cụm $\delta_{ext}(C_1) = 0.063 < \delta(G)$ nên khả năng cao C_1 là một cộng đồng.
- Với cụm C_2, ta có mật độ cụm $\delta_{int}(C_2) \approx 0.667 > \delta(G)$ và mật độ liên cụm $\delta_{ext}(C_2) \approx 0.133 < \delta(G)$ nên khả năng cao C_2 là một cộng đồng.
- Với cụm C_3, ta có mật độ cụm $\delta_{int}(C_3) = 0, \delta(G)$ và mật độ liên cụm $\delta_{ext}(C_1) \approx 0.333 < \delta(G)$ rất có thể C_3 không là một cộng đồng.

2.3.1 Trùng lặp

Một cá nhân có thể là thành viên của nhiều cộng đồng và cộng đồng này cũng có thể là một nhóm nhỏ của cộng đồng khác nên dẫn đến các câu trúc cộng đồng hết sức đa dạng như không trùng lặp (*non-overlap*), trùng lặp (*overlap*) hoặc chứa trong (*nest*). Hình 2.3 thể hiện ba dạng cấu trúc của cộng đồng. Đặc điểm và tính chất của các dạng cấu trúc cộng đồng trên như sau:

- **Cộng đồng không trùng lặp** là dạng các cộng đồng nằm tách biệt và không có cá nhân nào thuộc nhiều hơn 1 cộng đồng. Một số ví dụ về cộng đồng không trùng lặp như cộng đồng người Việt sống tại Việt Nam và cộng đồng người Việt sống tại châu Âu, cộng đồng người theo đạo Phật và cộng đồng người theo đạo Hồi, ... Tuy trên thực tế vẫn có trường hợp một người sống tại hai cộng đồng khác nhau hoặc một lúc theo hai tôn giáo nhưng trường hợp này rất ít và có thể xem là không trùng lặp.
- **Cộng đồng trùng lặp** là dạng cộng đồng phổ biến nhất và thường được bắt gặp trong thực tế nhất. Ví dụ như cộng đồng người thích chơi đá bóng và cộng đồng người thích chơi cầu lông. Sẽ có những người vừa thích chơi đá bóng và cầu lông và những người này được xem là các cá nhân trùng lặp của 2 cộng đồng trên, ngoài ra ta còn có một số ví dụ khác như cộng đồng dựa trên sở thích, nghề nghiệp,... cũng là các loại cộng đồng có trùng lặp.



Hình 2.3. Ba dạng cấu trúc của các cộng đồng.

- **Cộng đồng chứa trong cộng đồng** là một dạng đặc biệt của cộng đồng trùng lặp, khi có tồn tại một cộng đồng nào đó chứa trong một cộng đồng lớn hơn, Trường hợp này cũng rất phổ biến trong đời sống. Ví dụ như cộng đồng những người thích công nghệ và cộng đồng những người thích hệ điều hành Linux, cộng đồng những người thích bóng đá và cộng đồng những người thích chơi thể thao và cộng đồng những người thích bóng đá... Các trường hợp này thường xuất phát bởi các khái niệm nay bao quát khái niệm kia.

2.4 Vấn đề phát hiện cộng đồng trong mạng xã hội.

Phát hiện cộng đồng trên mạng xã hội là một vấn đề quan trọng và có nhiều ý nghĩa trong thực tiễn như giúp hiểu được các mối quan hệ tương tác, sự biến đổi của xã hội và dùng nó để giải quyết các vấn đề kinh tế, chính trị, xã hội,...

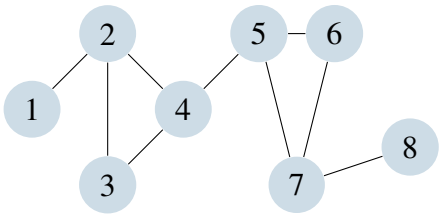
Có nhiều động lực của việc phát hiện cộng đồng khác nhau nên chúng ta cũng có những kiểu cộng đồng khác nhau. Ví dụ như trong một cộng đồng mạng xã hội, các nhà khoa học mong muốn phát hiện các cộng đồng để phục vụ cho việc quảng cáo, các nhà bán lẻ dụng cụ thể dục thể thao quan tâm đến các cộng đồng người yêu thích các môn thể thao; các nhàn bán lẻ điện thoại quan tâm đến các cộng đồng người thích đùng các thương hiệu nào như *Apple*, *Samsung* hay *Huawei*,... qua đó giúp tăng cường số lượng người nhận được quảng cáo mình cần, giúp tăng cường doanh thu. Trong các cuộc bầu chọn, các ừng cử viên có thể quan tâm đến các cộng đồng những người yêu thích, phản đối hay trung lập đối với mình để thực hiện các tác động như đưa tin, quảng cáo để thay đổi thái độ người bầu chọn giúp tăng cường số phiếu.

Bài toán phát hiện cộng đồng trên mạng xã hội là bài toán gom cụm người dùng trên mạng xã hội thành các cụm người dùng sao cho ta tìm được các cụm người dùng có cùng các đặc điểm mà ta mong muốn.

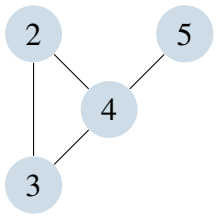
2.5 Đồ thị cục bộ và vấn đề phát hiện cộng đồng trên đồ thị cục bộ

Đồ thị cục bộ (*local graph*) là một đồ thị con của một đồ thị toàn cục (*global graph*) và nó tập trung để mô tả các mối quan hệ liên quan đến một hoặc một vài đỉnh. Trong một mạng xã hội,

đồ thị cục bộ được định nghĩa là đồ thị có tập đỉnh là một người dùng được xem như đỉnh *trung tâm*, bạn bè của họ được xem như các đỉnh *lân cận*. Tập cạnh sẽ là tập các mối quan hệ xung quanh những người dùng ở tập đỉnh. Đồ thị cục bộ này dùng để mô tả các mối quan hệ chính của người dùng *trung tâm* này. Hình 2.4 mô tả một ví dụ của mộ đồ thị cục bộ.



(a) Một đồ thị toàn cục với 8 đỉnh.



(b) Một đồ thị cục bộ của đỉnh 4 trong đồ thì toàn cục như Hình 2.4a.

Hình 2.4. Một ví dụ về đồ thị cục bộ.

Đồ thị cục bộ có một số tính chất như sau:

- Đỉnh *trung tâm* có cạnh nối đến tất cả các đỉnh còn lại.
- Tồn tại nhiều đỉnh có bậc bằng một. Điều đó nghĩa là ngoài đỉnh trung tâm ra, các đỉnh này không liên kết với các đỉnh nào khác trong đồ thị này cả. các đỉnh này gọi là *đỉnh đơn độc*.
- Các cạnh trên đồ thị cục bộ đều có một ý nghĩa nhất định đối với đỉnh trung tâm.

Công đồng ở mạng cục bộ có đặc điểm là mức độ trùng lặp ít và số lượng thành viên trong một cộng đồng có thể ít hoặc nhiều. Việc nghiên cứu và phát hiện cộng đồng trên đồ thị cục bộ mang đến một số lợi ích về việc phát hiện các cộng đồng đặc biệt đối với người dùng trung tâm như cộng đồng *gia đình*, *đồng nghiệp*, *bạn bè*, *bà con xa*, vân vân. Việc hiểu cấu trúc các cộng đồng trong đồ thị mạng cục bộ giúp ta hiểu thêm ở mức từng người dùng và có những ứng dụng tốt hơn trong một mạng xã hội truyền thông (*social media network*).

Chương 3

Các công trình liên quan

Trong chương này, chúng tôi trình bày tổng quan về các kết quả nghiên cứu gần đây có liên quan đến bài toán phát hiện cộng đồng.

3.1 Nhóm phương pháp dựa trên cơ sở Modularity

3.1.1 Độ đo modularity

Độ đo modularity được đề xuất bởi Newman and Girvan [3]. Nó được tính dựa trên sự phân của các cạnh bên trong một cộng đồng so với đồ thị ngẫu nhiên có cùng mật độ cạnh. Modularity có thể được phát biểu ở dạng

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\delta(C_i, C_j), \tag{3.1}$$

với tổng theo tất cả các cặp đỉnh. A là ma trận kề của đồ thị, m là tổng số cạnh của đồ thị, P_{ij} đại diện cho cho kỳ vọng của số cạnh giữa đỉnh thứ i và đỉnh thứ j. Hàm δ cho kết quả là một nếu đỉnh i và đỉnh j cùng một cộng đồng $(C_i = C_j)$ và ngược lại bằng không.

Trên thực tế, cạnh giữa hai đỉnh i và j là sự nối liền giữa hai *then* (nữa cạnh, tiếng Anh: stub). Xác suất p_i để chọn ngẫu nhiên một then và then đó thuộc i là $k_i/2m$, vì có k_i then ở đỉnh i và $2m$ then ở toàn bộ đồ thị. Xác suất của một kết nối giữa i và j được tính bằng tích $p_i p_j$, vì các cạnh được đặt độc lập với nhau. Kết quả là $k_i k_j/4m^2$, từ đó ta được kỳ vọng $P_{ij} = 2mp_i p_j = k_i k_j/2m$. Từ đó, ta có công thức tính modularity cuối cùng là

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j). \tag{3.2}$$

Vì tổng trên được đóng góp bởi các cặp cạnh trên cùng một cụm nên ta có thể viết lại ở dạng tổng của tổng các cạnh theo từng cụm trên tất cả các cụm

$$Q = \sum_{c}^{n_C}\left[\frac{l_c}{m} - \left(\frac{d_c}{2m}\right)^2\right]. \tag{3.3}$$

Ở đây, n_C là số lượng của các cụm, l_c là tổng số lượng các cạnh trong cụm c và d_c là tổng số lượng các bậc của các đỉnh trong c.

Ví dụ 3.1.1. Ma trận A sau là ma trận kề biểu diễn tương ứng với đồ thị ở Ví dụ 2.3.1:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Ta thực hiện tính modularity sử dụng công thức (3.3). Ta có:

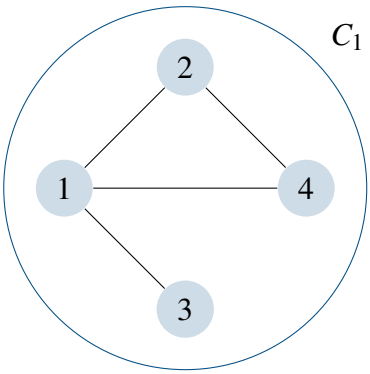
$$\begin{aligned} l_1 &= (\sum_{i \in C_1} (\sum_{j \in C_1} A_{ij}))/2 \\ &= ((A_{11} + A_{12} + ... + A_{14}) + (A_{21} + ... + A_{24}) + (...) + (A_{41} + ... + A_{44}))/2 \\ &= ((0+1+1+1) + (1+0+1+1) + (1+1+0+1) + (1+1+1+0))/2 \\ &= 6, \end{aligned}$$

$$\begin{aligned} d_1 &= \sum_{i \in C_1} (\sum_{j \in E} A_{ij}) \\ &= (A_{11} + A_{12} + ... + A_{18}) + (A_{21} + ... + A_{28}) + (...) + (A_{41} + ... + A_{48}) \\ &= ((0+1+1+1+0+0+0+0) + ... + (1+1+1+0+1+0+0+0) \\ &= 13, \end{aligned}$$

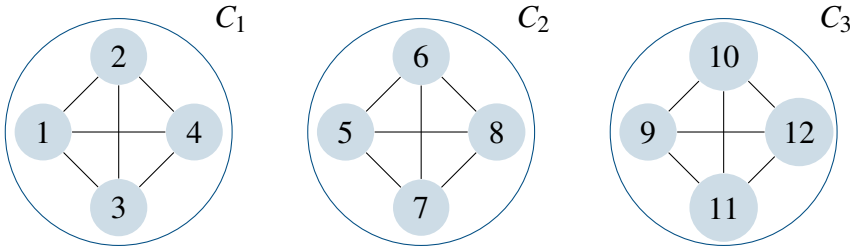
và tương tự ta được $l_2 = 2$, $d_2 = 6$, $l_3 = 0$, $d_3 = 4$. Thay các giá trị $l_1, l_2, l_3, d_1, d_2, d_3$ vào công thức (3.3) ta được:

$$\begin{aligned} Q &= \sum_{c}^{n_C} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] \\ &= \frac{l_1}{m} - \left(\frac{d_1}{2m} \right)^2 + \frac{l_2}{m} - \left(\frac{d_2}{2m} \right)^2 + \frac{l_3}{m} - \left(\frac{d_3}{2m} \right)^2 \\ &= \frac{6}{10} - \left(\frac{13}{20} \right)^2 + \frac{2}{10} - \left(\frac{6}{20} \right)^2 + \frac{0}{10} - \left(\frac{4}{20} \right)^2 \\ &= 0.31 \end{aligned}$$

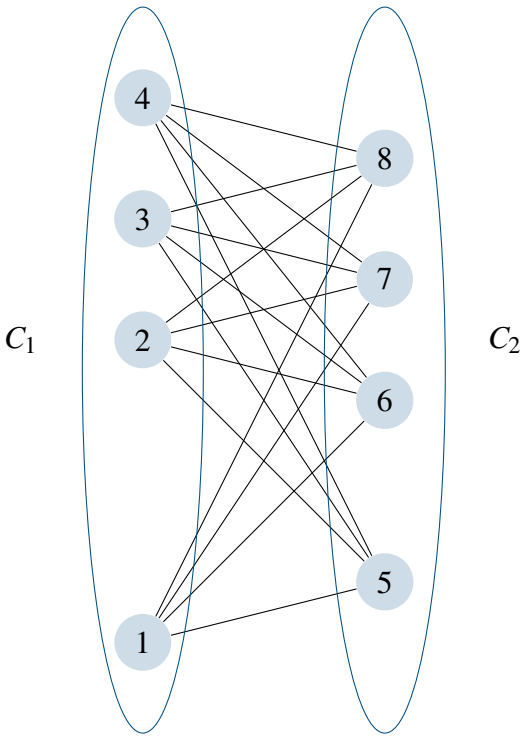
Ngoài ra ta có một số trường hợp đặc biệt của modularity đạt giá trị bằng 0, lớn nhất và nhỏ nhất như sau:



Hình 3.1. Đồ thị với cộng đồng bao phủ toàn bộ đồ thị.



Hình 3.2. Đồ thị không liên thông với các thành phần liên thông là các đồ thị đầy đủ (*complete graph*).



Hình 3.3. Đồ thị hai phía kết nối đầy đủ.

- **Trường hợp 1**: Cả đồ thị là một cộng đồng như Hình 3.1. Lúc này, $l_1 = m$ và $d_1 = 2m$ nên thay vào công thức (3.3) ta được giá trị modularity $Q = 0$.

- **Trường hợp 2**: Đồ thị không liên thông với các thành phần liên thông là các đồ thị đầy đủ (*complete graph*) như Hình 3.2. Lúc này, ta viết lại biểu thức tính modularity ở công thức (3.3) như sau:

$$Q = \sum_{c}^{n_C} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] = \sum_{c}^{n_C} \frac{l_c}{m} - \sum_{c}^{n_C} \left(\frac{d_c}{2m} \right)^2.$$

  Ta có $\sum_c^{n_C} \frac{l_c}{m} = 1$ vì $\sum_c^{n_C} l_c = m$. Mỗi thành phần $\left(\frac{d_c}{2m} \right)^2$ càng nhỏ nếu m càng lớn. Nghĩa là nếu ta tăng số lượng thành phần liên thông như Hình 3.2 tới vô cùng thì số m cũng tiến tới vô cùng và tổng sau sẽ xấp xỉ bằng 0. Từ đó ta được modularity sẽ tiến về 1.

- **Trường hợp 3**: Đồ thị hai phía kết nối đầy đủ như Hình 3.3. Lúc này $l_1 = 0$ và $l_2 = 0$ vì không có cạnh nào bên trong cộng đồng. $d_1 = d_2 = m$ nên tổng $\sum_c^{n_C} \left(\frac{d_c}{2m} \right)^2 = \left(\frac{m}{2m} \right)^2 + \left(\frac{m}{2m} \right)^2 = \frac{1}{2}$. Do đó, modularity sẽ đạt giá trị bằng $-\frac{1}{2}$.

3.1.2 Phương pháp Girvan-Newman

Phương pháp phát hiện cộng đồng nổi tiếng nhất là Girvan-Newman [3, 4] được đề xuất bởi Mark Newman và Michelle Girvan vào năm 2002. Thuật toán phát hiện cộng đồng bằng cách loại bỏ đi các cạnh khỏi mạng gốc. Các thành phần liên thông còn lại được kết nối với nhau trong mạng gốc sẽ là các cộng đồng. Các cạnh được chọn ở đây dựa theo *tính trung tâm cạnh* (*edge centrality*) và mức độ quan trọng của các cạnh được ước lượng dựa theo các thuộc tính của đồ thị đó. Các bước của thuật toán được mô tả như sau:

1. Tính *tính trung tâm* của tất cả các cạnh trong đồ thị.
2. Loại bỏ cạnh có *tính trung tâm* cao nhất.
3. Tính lại *tính trung tâm* của các cạnh bị ảnh hưởng bởi cảnh vừa loại bỏ.
4. Lặp lại bước 2 và 3 cho đến khi không có cạnh nào bị loại.

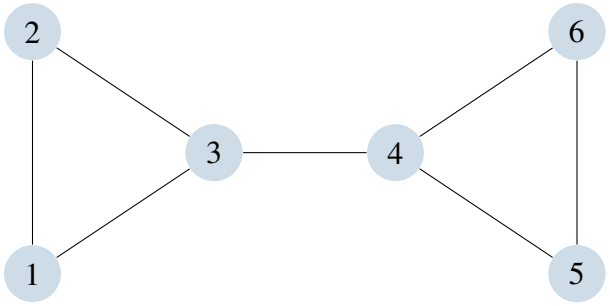
Thuật toán Girvan–Newman định nghĩa *độ giữa của cạnh* (*edge betweenness*) của một cạnh là số lượng đường đi ngắn nhất của các cặp cạnh chạy dọc qua nó. Trong một mạng, nếu các cộng đồng hoặc các nhóm được kết nối lỏng lẻo với nhau bởi một vài cạnh liên nhóm, thì tất cả các đường đi ngắn nhất giữa các cộng đồng khác nhau phải đi dọc theo một trong các cạnh này. Do đó, các cạnh kết nối cộng đồng sẽ có *độ giữa của cạnh* cao (ít nhất một trong số chúng). Bằng cách loại bỏ các cạnh này, các nhóm được tách ra khỏi nhau và do đó cấu trúc cộng đồng cơ bản của mạng sẽ được hiện ra.

Trên thực tế, việc tính lại *độ giữa của cạnh* sau mỗi lần loại bỏ một cạnh sẽ làm giảm thời gian chạy. Tuy nhiên, việc làm này là cần thiết để đáp ứng sự thay đổi của mạng sau khi một cạnh bị loại đi. Ví dụ, trong một mạng có hai cộng đồng và chúng được liên kết với nhau bằng nhiều hơn 1 cạnh thì các cạnh thì ta vẫn không thể đảm bảo được rằng tất cả các cạnh nối đó đều

có *độ giữa của cạnh* cao. Vì vậy, việc tính lại sau mỗi lần loại bỏ đi một cạnh là cần thiết để đảm bảo rằng một trong số các cạnh nối sẽ mang giá trị *độ giữa của cạnh* cao.

Kết quả cuối cùng của thuật toán Girvan – Newman là một *dendrogram* [5]. Khi thuật toán Girvan – Newman chạy, *dendrogram* được tạo ra từ trên xuống (tức là mạng chia tách thành các cộng đồng khác nhau với việc loại bỏ liên kết liên tiếp). Các lá của dendrogram là các nút riêng lẻ. Cuối cùng, ta dùng modularity để tìm nhát cắt thích hợp trên *dendrogram* để thu được cấu trúc cộng đồng có modularity cao nhất.

Ví dụ 3.1.2. Cho một đồ thị $G = (V,E)$ với $V = \{1,2,3,4,5,6\}$ và $E = \{(1,2), (1,3), (2,3), (4,3), (4,5), (4,6), (5,6)\}$ được mô tả như Hình 3.4. Ta áp dụng thuật toán Girvan – Newman để phát hiện cộng đồng cho đồ thị G.



Hình 3.4. Ví dụ về một đồ thị đơn.

Ta tìm đường đi ngắn nhất của tất cả các cặp đỉnh bằng giải thuật Dijkstra [6] và thu được kết quả danh sách các đường đi ngắn nhất như trong Bảng 3.1.

Cạnh	Đường đi ngắn nhất
(1,2)	$1 \to 2$
(1,3)	$1 \to 3$
(1,4)	$1 \to 3 \to 4$
(1,5)	$1 \to 3 \to 4 \to 5$
(1,6)	$1 \to 3 \to 4 \to 6$
(2,3)	$2 \to 3$
(2,4)	$2 \to 3 \to 4$
(2,5)	$2 \to 3 \to 4 \to 5$
(2,6)	$2 \to 3 \to 4 \to 6$
(3,4)	$3 \to 4$
(3,5)	$3 \to 4 \to 5$
(3,6)	$3 \to 4 \to 6$
(4,5)	$4 \to 5$
(4,6)	$4 \to 6$
(5,6)	$5 \to 6$

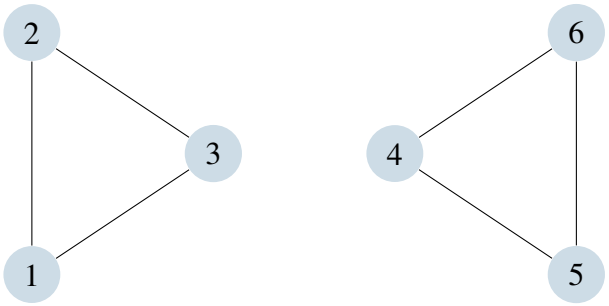
Bảng 3.1. Danh sách đường đi ngắn nhất của các cặp cạnh của đồ thị trong Hình 3.4.

Ta *tính độ giữa của cạnh* của tất cả các cạnh bằng cách tính số lượng đường đi ngắn nhất đi qua nó. Kết quả thu được như Bảng 3.2.

Cạnh	Đường đi ngắn nhất
(1,2)	1
(1,3)	4
(2,3)	4
(3,4)	9
(4,5)	4
(4,6)	4
(5,6)	1

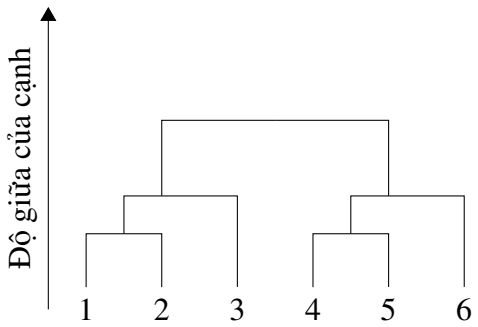
Bảng 3.2. Giá trị *độ giữa của cạnh* của các cạnh dựa vào Bảng 3.1.

Từ đó, ta chọn cạnh có *độ giữa của cạnh* cao nhất là cạnh $(3,4)$ và xóa cạnh đó đi. Hình 3.5 thể hiện đồ thị sau khi đã xóa đi cạnh $(3,4)$. Sau khi loại bỏ cạnh $(3,4)$ trong đồ thị, sô thành phần liên thông tăng lên thành 2 thành phần liên thông là C_1 với các đỉnh là $\{1,2,3\}$ và thành phần liên thông C_2 với các đỉnh là $\{4,5,6\}$. Lúc này, C_1 C_2 có thể được xem là các cộng đồng trong đồ thị G.



Hình 3.5. Đồ thị ở Hình 3.4 sau khi xóa cạnh (3,4).

Lặp lại các bước làm trên cho đến khi mỗi đỉnh là một thành phần liên thông (tất cả các cạnh đã bị xóa) ta thu được dendrogram như Hình 3.6



Hình 3.6. Dendrogram sau khi dùng giải thuật Girvan-Newman trên đồ thị G.

Với kết quả trên, ta thực hiện xem xét chọn nhát cắt để thu được giá trị modularity cao nhất. Với trường hợp chọn 2 thành phần liên thông như ở Hình 3.5 ta thu được giá trị modularity cao nhất là $Q \approx 0.357$.

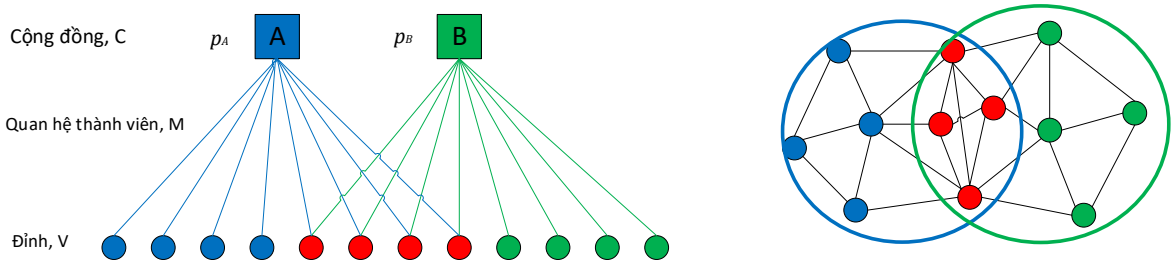
3.2 Nhóm phương pháp dựa trên Factorize matrix

Trong phần này, ta sẽ trình bày về cách mô hình bài toán về dạng mô hình AGM và giải thuật BigCLAM, phương pháp phát hiện cộng đồng có trùng lặp trên quy mô lớn hàng triệu đỉnh.

3.2.1 Mô hình đồ thị liên kết

Mô hình đồ thị liên kết (Affiliation Graph Model - AGM)[7–9] là một mô hình sinh $B(V,C,M,\{p_c\})$ cho đồ thị với:

- Ta có số lượng các cộng đồng và số lượng các cá nhân (các đỉnh trong đồ thị).
- Bất kỳ cá nhân nào cũng có thể là thanh viên của một cộng đồng. Qua đó ta có quan hệ thành viên (membership) M trong mỗi cộng đồng là một tham số (parameter) của mô hình.
- Mỗi cộng đồng C có một xác suất p_c có liên quan tới nó là xác suất hai cá nhân thuộc cộng đồng C được kết nối bởi một cạnh. Xác suất này cũng là một tham số của mô hình.
- Nếu một cặp đỉnh trong 2 hoặc nhiều cộng đồng thì có một cạnh giữa chúng với xác suất được tính theo công thức (3.4)



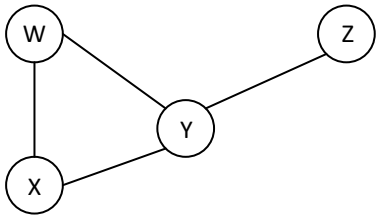
Hình 3.7. Mô hình đồ thị liên kết (AGM) trái và mạng tương ứng (phải).

Chúng ta phải tính độ hợp lý likelihood của một đồ thị với số lượng đỉnh thích hợp được tạo ra theo cơ chế này. Vấn đề chính ở đây là làm thế nào để tính được xác suất của một cạnh khi biết cá nhân đó thuộc cộng dồng nào và các giá trị của p_c. Xét một cạnh (u,v) giữa đỉnh hai đỉnh u và v. Giả sử u và v là thành viên của các cộng đồng C và D và không thuộc các cộng đồng khác. Xác suất không có cạnh giữa u và v là phép nhân *product* của xác suất để không có cạnh ở cộng đồng C và xác suất để không có cạnh ở cộng đồng D. Nó là xác suất $(1-p_C)(1-p_D)$ nên từ đó suy ra xác suất để nó là một cạnh là $1-(1-p_C)(1-p_D)$.

Một cách tổng quát, nếu u và v là thành viên của một tập khác rỗng các cộng đổng M và không thuộc các cộng đồng khác thì p_{uv} là xác suất của một cạnh giữa u và v được tính bởi:

$$p_{uv} = 1 - \prod_{C \text{ in } M} (1-p_C) \tag{3.4}$$

Ngoài ra, khi u và v không cùng nằm trong bất kỳ cộng đồng nào thì $p_{uv} = \varepsilon$ là một số thực dương rất bé. Ta chọn để xác suất luôn lớn hơn không.



Hình 3.8. Đồ thị mạng xã hội đơn giản.

Mô hình AGM có thể mô tả được linh hoạt các dạng cộng đồng như phân tách, chồng lặp hay lồng vào nhau.

Nếu ta biết tất cả các đỉnh thuộc về những cộng đồng nào thì ta có thể tính được độ hợp lý likelihood của đồ thị bằng cách dựa vào công thức trên. Cho M_{uv} là tập cộng đồng mà u và v cùng thuộc. Ta có likelihood của E (tập cạnh của đồ thị) là

$$\prod_{(u,v)\ in\ E} p_{uv} \prod_{(u,v)\ not\ in\ E} (1-p_{uv}) \tag{3.5}$$

Ví dụ: Xét một mạng xã hội đơn giản ở Hình 3.8. Giả sử có hai cộng đồng là C và D với xác suất p_C và p_D và $C = \{w,x,y\}$ và $D = \{w,y,z\}$. Từ đó, ta bắt đầu xét cặp đỉnh w và x. $M_{xw} = C$ vì đây là cặp đỉnh nằm trong C nhưng không nằm trong D. Do đó $p_{xw} = 1-(1-p_C) = p_C$.
Tương tự, với x và y chỉ cùng cộng đồng C, y và z chỉ cùng cộng đồng D, w và z cùng cộng đồng D. Từ đó ta tìm được $p_{xy} = p_C$ và $p_{yz} = p_{wz} = p_D$. w và y nằm cùng hai cộng đồng C và D nên $p_{wy} = 1-(1-p_C)(1-p_D) = p_C + p_D - p_C p_D$. Cuối cùng x và z không nằm cùng một cộng đồng nên $p_{xz} = \varepsilon$.
Với giả định như trên ta có thể tính được likelihood của đồ thị 3.8 bằng công thức 3.5. Ta được

$$p_{wx}p_{wy}p_{xy}p_{yz}(1-p_{wz})(1-p_{xz}) \tag{3.6}$$

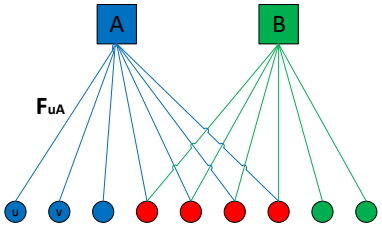
Thay thế các biểu thức ta đã tính được ở trên ta được

$$(p_C)^2 p_D(p_C + p_D - p_C p_D)(1-p_D)(1-\varepsilon) \tag{3.7}$$

Vì ε rất nhỏ nên $(1-p_{xz})$ có thể làm tròn thành 1. Ta cần tìm giá trị của p_C và p_D sao cho biểu thức 3.7 đạt giá trị lớn nhất. Chú ý rằng mỗi nhân tử trong biểu thức trên hoặc là độc lập với p_C hoặc là tăng theo p_C. Đồng thời ta cũng nhớ rằng $p_D \leq 1$. Ta có

$$p_C + p_D - p_C p_D \tag{3.8}$$

tăng theo p_C nên hợp lý cực đại (maximun likelihood) khi p_C lớn nhất có thể nên $p_C = 1$. Với $p_C = 1$, biểu thức còn lại là $p_D(1-p_D)$ đại cực đại khi $p_D = 0.5$. Từ đó, cho $C = \{w,x,y\}$ và $D = \{w,y,z\}$, hợp lý cực đại của đồ thị 3.8 đạt được khi giữa các đỉnh trong C chắc chắn có tồn tại cạnh giữa chúng và các giữa đỉnh trong D có 50% khả năng tồn tại cạnh giữa chúng.



Hình 3.9. Độ mạnh mối liên kết đối với cộng đồng.

Ở ví dụ, trên ta chỉ giải quyết được một phần của bài toán. Chúng ta cần tìm phép gán thành viên đến cộng đồng nào sao cho đạt được hợp lý cực đại. Để giải bài toán tối ưu kiểu này ta có thể sử dụng giải thuật gradient descent[10, 11]. Tuy nhiên vì một cá nhân chỉ liên kết với một cộng đồng qua một mối qua hệ rời rạc (có hoặc không). Không thể sử dụng một hàm số liên tục để gradient descent thực hiên tối ưu được. Các duy nhất là ta có thể thực hiện một phép gán và lần lượt tạo các thay đổi nhỏ trong các phép gán các thành viên với các cộng đồng. Tuy nhiên cách này không chắc sẽ tìm được phép gán tốt nhất.

3.2.2 Giải pháp tránh trường hợp mối quan hệ thành viên liên tục

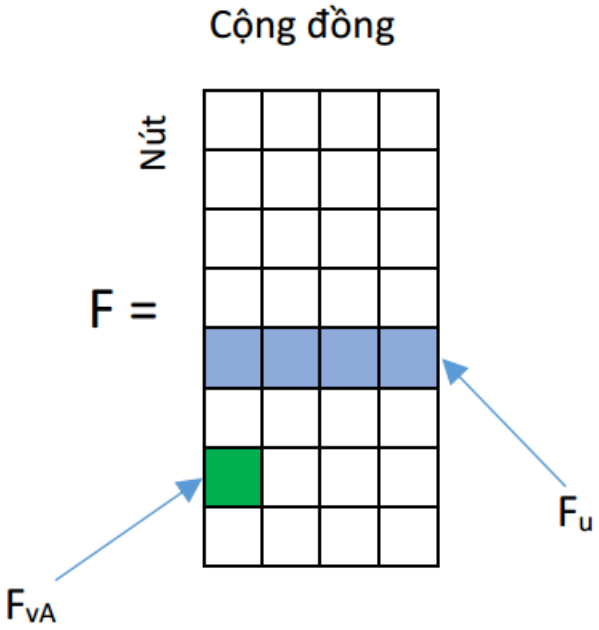
Trên thực tế có những cá nhân có mối liên kết mạnh với một số cộng đồng và yếu hơn với một số khác chứ không chỉ ở mức rời rạc có hoặc không. Ta có một độ đo cho mối liên kết giữa một cá nhân với một cộng đồng đó là "độ mạnh quan hệ thành viên" ("strength of membership") của mỗi cá nhân với mỗi cộng đồng. Theo trực giác ta đánh giá nếu hai cá nhân có độ mạnh quan hệ thành viên đến một cộng đồng lớn thì khả năng cao cạnh giữa họ là do sự ảnh hưởng từ cộng đồng này. Trong mô hình này, ta có thể điêu chính độ mạnh của quan hệ thành viên ở dạng liên tục cũng như xác suất liên quan với một cộng đồng trong mô hình đồ thị liên kết. Điều này giúp ta có thể sử dụng gradient descent để cực đại hóa biểu thức likelihood. Trong mô hình cải tiếng này ta

- Cố định các tập cộng đồng và cá thể.
- Với mỗi cộng đồng C và cá nhân x ta có một tham số độ mạnh quan hệ thành viên F_{xC}. Tham số này là một số không âm với trường hợp bằng 0 nghĩa là cá nhân này không thuộc cộng đồng này.
- Xác suất cạnh giữa u và v được tạo ra bởi cộng đồng C là

$$p_C(u,v) = 1 - e^{-F_{uC}F_{vC}} \tag{3.9}$$

Hình 3.10 mô tả một ma trận F với |C| cột và |V| dòng. F_vA là độ mạnh quan hệ thành viên giữa cá nhân v và cộng đồng A và F_u là vector độ mạnh quan hệ thành viên với các cộng đồng.

Xác suất để có một cạnh giữa u và v sẽ là 1 trừ xác suất của không có cạnh nào giữa u và v mà xác suất để không có cạnh nào giữa u và v được tính bằng tích các xác suất không có cạnh



Hình 3.10. Ma trận F với |C| cột và |V| hàng.

nào giữa u và v trong mỗi cộng đồng. Từ đó ta có công thức tính p_{uv} như sau:

$$p_{uv} = 1 - \prod_{C}(1 - p_C(u,v)) \tag{3.10}$$

Ta có thể xem xác suất này là xác suất có ít nhất một cộng đồng chung. Biến đổi biểu thức 3.10

$$\begin{aligned} p_{uv} =& 1 - \prod_{C}(1 - p_C(u,v)) \\ =& 1 - e^{-\Sigma_C F_{uC} F_{vC}} \\ =& 1 - e^{-F_u F_v} \end{aligned} \tag{3.11}$$

Ví dụ 3.2.1. Cho F như Bảng 3.3:
Ta tính được: $F_u.F_v^T = 0.16$

F_u:	0	1.2	0	0.2
F_u:	0.5	0	0	0.8
F_u:	0	1.8	1	0

Bảng 3.3. Ma trận F đơn giản

Và $P(u,v) = 1 - e^{-0.16} = 0.14$
Tương ta được: $P(u,w) = 0.88$
$P(wv) = 0$

3.2.3 Giải thuật BigCLAM

Cho tập E là ta tập các cạnh trong đồ thị đang xét thì ta có thể viết công thức tính likelihood của đồ thị bằng cách lấy tích *product* của các biểu thức p_{uv} với (u,v) thuộc E và nhân với tích *product* của $(1-p_{uv})$ với (u,v) không thuộc E. Như vậy, công thức tính likelihood của mô hình mới này sẽ là

$$\prod_{(u,v)\in E}(1-e^{-\sum_C F_{uC}F_{vC}})\prod_{(u,v)\notin E}(e^{-\sum_C F_{uC}F_{vC}}) \tag{3.12}$$

Để đơn giản hóa biểu thức 3.5 ta lấy logarithm cơ số e của biểu thức để các dấu tích chuyển thành tổng và triệt tiêu hàm mũ. Lưu ý rằng khi lấy cực đại của likelihood cũng đồng nghĩa với việc lấy cực đại của logarithm của biểu thức đó.

$$l(F)=\sum_{(u,v)\in E} log(1-e^{-\sum_C F_{uC}F_{vC}})-\sum_{(u,v)\notin E}\sum_C F_{uC}F_{vC}. \tag{3.13}$$

Biểu thức log-likelihood 3.13 đạt được cực đại phụ thuộc vào giá trị của F_xC. Cách đơn giản nhất là sử dụng gradient descent với gradient theo F_u là

$$\nabla l(F_u)=\sum_{v\in N(u)} F_v\frac{e^{-F_uF_v^T}}{1-e^{-F_uF_v^T}}-\sum_{v\notin N(u)} F_v \tag{3.14}$$

Với $N(u)$ là tập các đỉnh kề với đỉnh u.

Ngoài ra ta có thể tăng tốc quá trình tính toán bằng cách tính

$$\sum_{v\notin N(u)} F_v=(\sum_v F_v-F_u-\sum_{v\in N(u)} F_v) \tag{3.15}$$

Bây giờ việc tính $\sum_{v\notin N(u)} F_v$ chỉ mất thời gian tuyến tính theo $|N(u)|$. Trên thực tế $|N(u)|$ nhỏ hơn rất nhiều so với $|V|-|N(u)|$.

Chương 4

Phương pháp đề xuất

4.1 Định nghĩa bài toán

Chúng ta sẽ mô hình hóa cụ thể vấn đề mở Mục 2.4 thành bài toán cụ thể như bên dưới.

- Dữ liệu đầu vào: Cho một đồ thị cục bộ là một đồ thị đơn $G = (V, E)$ với V là tập đỉnh, E là tập cạnh với $V \neq \emptyset, E \neq \emptyset$ và $\mathscr{M}, \mathscr{L}, \mathscr{C}$ lần lược là danh sách số lượng tin nhắn, danh sách số lượng lược thích bài đăng, danh sách số lượng lược danh sách số lượng lược bình luận bài đăng của các cặp người dùng trên mạng với $\mathscr{M}, \mathscr{L}, \mathscr{C} \geq 0$.
- Kết quả đầu ra: Danh sách n cộng đồng $C = \{C_1, C_2, ...C_n\}$ với mỗi cộng đồng là tập hợp của các đỉnh trên đồ thị G với $n > 0$, $C_i \neq \emptyset$ và $C_1 \vee C_2 \vee ... \vee C_n = V$.

4.2 Độ đo trên sự tương tác

Hầu hết các bộ dữ liệu về mạng xã hội hiện tại điều được mô hình dưới dạng đồ thị không trọng số. Điều đó nghĩa là chỉ có hai loại mối quan hệ của các người dùng trên mạng xã hội là có quan hệ hoặc không quan hệ (quan hệ có thể là bạn bè đối với Facebook, Zalo hoặc theo dõi đối với Twitter, Instargram). Trên thực tế, ta có nhiều loại mối quan hệ giữa hai người dùng vì chịu ảnh hưởng từ xã hội thực và mức độ thân thiết của hai người đùng trên mạng xã hội cũng khác nhau. Vì vậy, một đồ thị có trọng số để thể hiện mức độ kết nối giữa hai người dùng sẽ có ý nghĩa trong việc phân tích cộng đồng hơn là một đồ thị không trọng số.

Lấy cảm hứng từ độ đo *FLEX* [12, 13], chúng tôi định nghĩa một độ đo mới có dựa vào độ tương tác của các người dùng trên mạng xã hội dùng để tính trọng số các cạnh của một đồ thị. Ta bắt đầu tính trọng số của cạnh đặc trưng cho độ gắng kết các cộng đồng trong đồ thị dựa vào sự tương tác của các người dùng trong mạng xã hội. Cho hai đỉnh i và j với $i \neq j$. Ta định nghĩa $F(i, j)$ là *độ kết nối trực tiếp* giữa i và j. $F(i, j)$ được tính bằng công thức

$$F(i, j) = E_{ij}, \tag{4.1}$$

với E_{ij} là cạnh nối giữa hai đỉnh i và j. Nếu tồn tại cạnh giữa i và j thì $F(i, j) = E_{ij} = 1$ và ngược lại, nếu không tồn tại cạnh giữa i và j thì $F(i, j) = E_{ij} = 0$.

Độ kết nối gián tiếp $G(i,j)$ giữa hai đỉnh i và j được tính bằng công thức

$$G(i,j) = \frac{2|\Gamma_i \wedge \Gamma_j| - 2E_{ij}}{|\Gamma_i| + |\Gamma_j| - 2E_{ij}}, \tag{4.2}$$

trong đó Γ_i, Γ_j lần lược là tập các đỉnh lân cận với đỉnh i, j; $\Gamma_i \wedge \Gamma_j$ được gọi là các *đỉnh cầu* của i và j.

Độ tương tác giữa i và j là $H(i,j)$ được tính bằng công thức

$$H(i,j) = \gamma_{(1)}\mathscr{M}_{ij} + \gamma_{(2)}\mathscr{C}_{ij} + \gamma_{(3)}\mathscr{L}_{ij}, \tag{4.3}$$

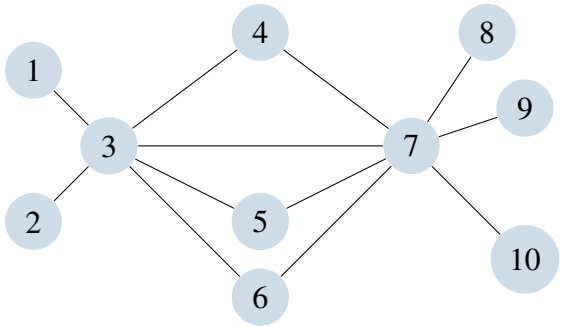
với $\gamma_{(1)}, \gamma_{(2)}, \gamma_{(3)}$ là các tham số; $\mathscr{M}_{ij}$ là số lượng tin được giửi qua lại giữa i và j; $\mathscr{C}_{ij}$ và $\mathscr{L}_{ij}$ lần lược là số lượng bình luận và thích (trên các bài đăng, ảnh, trạng thái,...) được thực hiện qua lại giữa i và j.

Dựng dựa vào (4.1), (4.2), (4.3), độ đo $D(i,j)$ được tính bằng công thức

$$D(i,j) = \alpha F(i,j) + (1-\alpha)G(i,j) - \frac{\beta}{H(i,j)}, \tag{4.4}$$

với α, β là các tham số, $\alpha, \beta \in [0,1]$. Độ đo $D(i,j)$ thể hiện trọng số của một cạnh có tỷ lệ với độ kết nối trực tiếp, độ kết nói gián tiếp và mức độ tương tác giữa các người dùng với nhau. Tham số α thể để điều chỉnh mức độ qua trọng của kết nối trực tiếp so với gián tiếp. Tham số α càng lớn thì độ kết nối trực tiếp có mức độ quan trọng lớn hơn so với độ kết nối gián tiếp, $\frac{\beta}{H(i,j)}$ được xem như điểm trừ của việc ít tương tác. Do đó, nếu β càng lớn thì điểm trừ này càng lớn.

Ví dụ 4.2.1. Cho một đồ thị như Hình 4.1. Ta xét hai đỉnh $i = 3$ và đỉnh $j = 7$ có độ tương tác như Bảng 4.1.



Hình 4.1. Một đồ thị đơn.

Cạnh	Số lượng tin nhắn	Số lượng thích	Số lượng bình luận
Cạnh (3,7)	150	2	10

Bảng 4.1. Bảng thể hiện mức độ tương tác của môt số người dùng trong mạng ở Hình 4.1 trong một tuần.

Cho $\alpha = 0.8, \beta = 1.0$ và $\gamma_1 = 0.1, \gamma_2 = 1, \gamma_3 = 0.5$ Ta thực hiện tính $D(i,j)$ như sau. Ta có ma trận kề A biểu diễu cho đồ thị ở Hình 4.1

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Độ kết nối trực tiếp giữa hai đỉnh i và j là

$$F(i,j) = E_{ij} = A_{ij}. \tag{*}$$

Ta có

$$\Gamma_i = \{1,2,4,5,6,7\}$$

và

$$\Gamma_j = \{4,5,6,7,8,9,10\}.$$

Từ đó, tính được

$$\Gamma_i \wedge \Gamma_j = \{4,5,6,7\}.$$

$$\Rightarrow |\Gamma_i \wedge \Gamma_j| = 4.$$

Độ kết nối gián tiếp của hai đỉnh i và j được tính là

$$G(i,j) = \frac{2|\Gamma_i \wedge \Gamma_j| - 2E_{ij}}{|\Gamma_i| + |\Gamma_j| - 2E_{ij}} = \frac{2\times 4 - 2\times 1}{6+7-2\times 1} = \frac{6}{11}. \tag{**}$$

Ta tính được độ tương tác giữa i và j là

$$\begin{aligned} H(i,j) &= \gamma_{(1)}\mathscr{M}_{ij} + \gamma_{(2)}\mathscr{C}_{ij} + \gamma_{(3)}\mathscr{L}_{ij} \\ &= 0.1\times 150 + 1\times 2 + 0.5\times 10 \\ &= 22. \end{aligned} \tag{***}$$

Thay (*), (**), (***) và công thức (4.4) ta tính được

$$\begin{aligned} D(i,j) &= \alpha F(i,j) + (1-\alpha)G(i,j) - \frac{\beta}{H(i,j)} \\ &= 0.8 \times 1 + (1-0.8) \times \frac{6}{11} - \frac{1}{22} \\ &= \frac{19}{22} \approx 0.86 \end{aligned}$$

Vậy từ đó, ta gán được trọng số của cạnh $(3,7)$ là 0.86.

4.3 Mô hình gom cụm

Phương pháp đề xuất được dựa trên mô hình gom cụm của giải thuật Louvain [14] với một số thay đổi để cải tiến quá trình tính toán và độ kết quả gom cụm. Ý tưởng của chúng tôi là gọi "đệ quy lại" trong cụm đỉnh đơn để phát hiện thêm các cộng đồng nhỏ khác.

4.3.1 Giải thuật Louvain

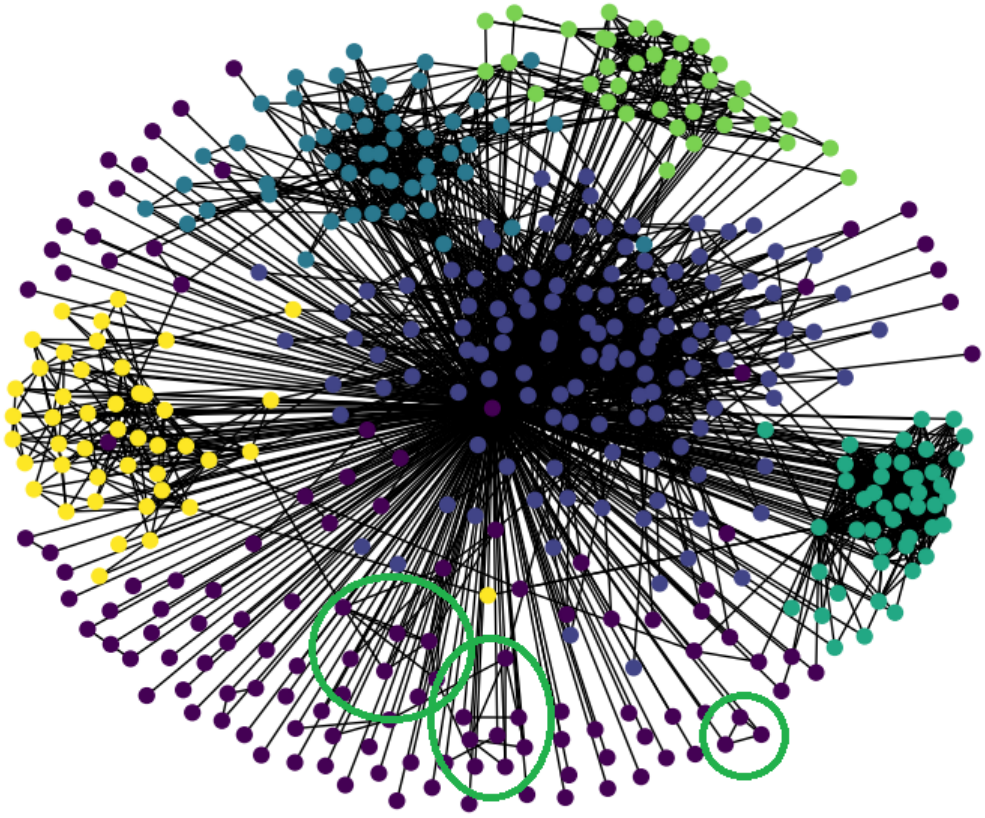
Modularity là metric định lượng độ tốt của việc gán các đỉnh cho cộng đồng bằng cách đánh giá kết nối dày đặc hơn các đỉnh trong một cộng đồng như thế nào so với cách kết nối trung bình trong một mạng ngẫu nhiên được xác định phù hợp. Phương pháp phát hiện cộng đồng Louvain là một thuật toán để phát hiện các cộng đồng trong các mạng dựa trên phương pháp heuristic để tối đa hóa modularity. Giải thuật Louvain lặp lại hai bước chính. Bước đầu tiên là gán *tham lam* (greedy) của các đỉnh cho một cộng đồng, nó ưu tiên tối ưu cục bộ giá trị modularity. Do đó, nó cần khởi tạo cho mỗi đỉnh một cộng đồng. Sau đó, với mỗi đỉnh i, ta lần lược xét các lân cận j của i và tính toán giá trị modularity sau khi xóa đỉnh i ra khỏi cộng đồng của nó và chuyển qua cộng đồng cùng với j. Đỉnh i tiếp theo sẽ được đặt vào cùng cộng đồng có giá trị modularity cao nhất và dương (lớn hơn 0). Nếu độ tăng của modularity không lớn hơn không, đỉnh i sẽ ở lại cộng đồng ban đầu.

Độ biến thiên của modularity ΔQ thu lại từ việc duy chuyển một đỉnh cô lập i đến cộng đồng C có thể được tính bằng công thức:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right], \tag{4.5}$$

với $\sum_{in}$ là tổng trọng số các cạnh bên trong C, $\sum_{tot}$ là tổng trọng số các cạnh được nối đến C, k_i là tổng trọng số của các cạnh nối đến đỉnh i, $k_{i,in}$ là tổng trọng số các cạnh từ i nối đến cộng đồng C và m là tổng số các cạnh trong đồ thị.

Bước thứ hai là xây dựng một đồ thị mới với một đỉnh bây giờ là các cộng đồng được tìm thấy trong giai đoạn đầu tiên. Trọng số của hai đỉnh lúc này được tính bằng tổng trọng số của các cạnh giữa hai cộng đồng trước đó [15]. Các cạnh giữa các đỉnh trong cùng một cộng đồng sẽ trở



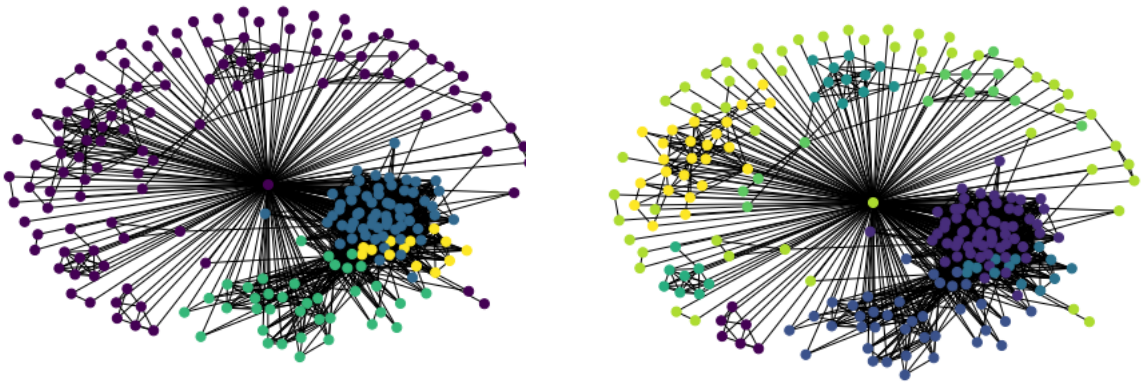
Hình 4.2. Phát gom cụm cộng đồng với Louvain. Vẫn còn một số cụm nhỏ (được khoanh tròn màu xanh lá) bị đẩy vào chung với cụm chưa node đơn lẻ.

thành khuyên (loop) trong đồ thị mới. Khi bước thứ hai kết thúc, ta có thể lặp lại bước một với đồ thị có trọng số.

Hai bước này được lặp lại cho đến khi modularity không thể tăng thêm và đạt giá trị tối đa. Kết quả modularity của mô hình gom cụm sử dụng phương pháp Louvain tốt tương đương khi so sánh với các thuật toán có sẵn, đồng thời thời gian chạy thấp hơn, vì vậy nó cho phép nghiên cứu các đồ thị lớn. Nó cũng thường cho thấy một hệ thống phân cấp của các cộng đồng ở các quy mô khác nhau, và quan điểm phân cấp này có thể hữu ích để hiểu được chức năng toàn cầu của một mạng.

4.3.2 Cải tiến giải thuật Louvain

Giải thuật Louvain giải quyết khá nhanh và tốt bài toán gom cụm đồ thị, tuy nhiên đối với đồ thị cục bộ (local graph) của các mạng xã hội có tính riêng tư cao (như các mạng xã hội nhắn tin) các cụm lớn thường được phát hiên dễ dàng nhưng các cụm nhỏ thường không được phát hiện và được đẩy vào cùng các *đỉnh đơn độc* (đỉnh bậc một). Hình 4.2 thể hiện một đồ thị cục bộ sau khi được phát hiện cộng đồng. Các đỉnh nằm cùng cộng đồng sẽ có cùng màu. Bằng trực giác, ta thấy các cụm được gom khá tốt, đồ phân bố các cạnh trong một cộng đồng cao hơn so với trên toàn đồ thị số cạnh kết nối qua lại giữa các cụm không nhiều. Tuy nhiên vẫn còn một số cụm



(a) Một đồ thị cục bộ sau khi được phát hiện cộng đồng với giải thuật Louvain. Nhiều cụm nhỏ vẫn không được phát hiện và gộp cùng với cụm *đỉnh đơn độc*.

(b) Một đồ thị cục bộ sau khi được phát hiện cộng đồng với giải thuật Louvain cải tiến. Các cụm nhỏ đã được phát hiện và tách riêng với cụm *đỉnh đơn độc*.

Hình 4.3. Khác nhau giữa kết quả gom cụm bằng giải thuật Louvain (trái) và Louvain cải tiến (phải).

nhỏ được khoanh tròn màu xanh lá vẫn chưa được phát hiện. Các cụm này tuy có kích thước nhỏ nhưng vẫn mang lại ý nghĩa rất lớn trong thực nghiệm. Từ đó, ta đề xuất một ý tưởng cái tiến giải thuật là sau khi phát hiên cộng đồng, ta thực hiện tìm kiếm cộng đồng chứa các *đỉnh đơn độc* và thực hiện phát hiện cộng đồng một lần nữa trên cộng đồng này.

Hình 4.3 cho ta thấy sự khác biệt giữa hai phương pháp Louvain và Louvain cải tiến. Với phương pháp Louvain cải tiến, các cụm nhỏ được phát hiện và tách riêng so với cụm *đỉnh đơn độc*.

4.3.3 Cấu trúc dữ liệu

Trong phần này, chúng tôi sẽ trình bày về cấu trúc dữ liệu để lưu trữ giúp tăng tốc quá trình tính toán và giảm không gian lưu trữ dữ liệu. Có hai loại cấu trúc dữ liệu phổ biến để thể hiện một đồ thị:

- Ma trận kề
- Danh sách kề

Bảng 4.2. So sánh hai cách biểu diễn đồ thị, V là danh sách cạnh, E là danh sách đỉnh, K tập các cạnh nối đến với đỉnh u. Ta giả định rằng $|V| < |E|$.

	Ma trận kề	Danh sách kề
Bộ nhớ lưu trữ	$O(\|V\|^2)$	$\boldsymbol{O(\|E\|)}$
Thêm một cạnh mới	$\boldsymbol{O(1)}$	$O(1)$
Xóa một cạnh	$\boldsymbol{O(1)}$	$O(\|K\|)$
Tìm một cạnh	$\boldsymbol{O(1)}$	$O(\|K\|)$
Liệt kê các cạnh kề với một đỉnh u	$O(\|V\|)$	$\boldsymbol{O(\|K\|)}$

Với mỗi loại cấu trúc dữ liệu khác nhau, ta có những ưu điểm và nhược điểm khác nhau. Bảng 4.3 so sánh độ phức tạp của hai loại cấu trúc dữ liệu ma trận kề và danh sách kề.

Tính toán với một đồ thị được biểu diễn bởi ma trận kề thì rất nhanh. Tuy nhiên, nếu đồ thị quá lớn thì chúng ta không thể sử dụng một ma trận lớn để biểu diễn đồ thị nếu tài nguyên bố nhớ của ta có hạn. Lúc này, danh sách kề là sự lựa chọn tốt hơn khi chỉ cần $O(|E|)$ không gian bộ nhớ thay vì $O(|V|^2)$ và liệt kê các cạnh kề với một đỉnh cũng nhanh hơn so với ma trận kề.

Trong luận văn này, chúng tôi sử dụng một cấu trúc dữ liệu khác đó là *HashList* [16] được đề xuất bởi MA Kolosovskiy. Đồng thời, có thay đổi các tham số như khởi tạo độ lớn của mảng băm (hash table) để phù hợp với dữ liệu hơn.

Biểu diễn đồ thị với danh sách liên kết

Danh sách liên kết (linked list) là kiểu dữ liệu tuyến tính, trong đó các phần tử là một đối tượng riêng biệt. Với danh sách liên kết, quá trình thêm và xóa một phần tử mất chỉ mất $O(1)$, quá trình tìm kiếm và truy cập đến một phần tử mất $O(n)$.

Việc sử dụng danh sách liên kết để biểu diễn đồ thị không quá phức tạp, ta sử dung một mảng để lưu các con trỏ (pointer) của các danh sách liên kết đại diện cho các đỉnh. khi Truy xuất tới một cạnh, ta truy cập tới đỉnh đó để lấy con trỏ cho danh sách cạnh. Lúc này, ta thực hiện các bước tìm kiếm, thêm, xóa các cạnh trên danh sách liên kết đã được chọn.

Mã giả của việc biểu diễn đồ thị sử dụng danh sách liên kết được mô tả như sau:

```
declare user, head[], next[], data[]
procedure createMultiList(numVertices, numEdges):
    heads = int[numVertices]
    next = int[numEdges + 1]
    data = int[numEdges + 1]

procedure add(x,y):
    used += 1
    data[used] = y
    next[used] = heads[x]
    heads[x] = used

boolean contains(x,y):
    for ( i = heads[x] ; i <> 0 ; i = next[i]):
        if data[i] == y:
            return true;
    return false
```

Biểu diễn đồ thị với bảng băm

Bảng băm (hash table) có ưu điểm là thực hiện tìm kiếm một phần tử với thời gian $O(1)$. Vì vậy, ta cân nhắc đến việc sử dụng hàm băm để lưu trữ cạnh. Có hai khó khăn của phương pháp này 1 là cần tìm ra một hàm băm thích hợp cho việc băm với 2 số. Và khó khăn thứ 2 là ta không thể

xóa một phần tử trong bảng băm được, tuy nhiên, ta có thể đánh dấu cạnh khi bị xóa bằng một giá trị đặc biệt.

Mã giả của việc biểu diễn đồ thị sử dụng bảng băm được mô tả như sau:

```
declare SIZE = 1000000, data[], used
procedure add(x,y):
    code = code(x,y)
    hash = hash(x,y)
    while used[hash]:
        if data[hash] == code:
            return false
        else:
            return hash = (hash+1) mod SIZE
    used[hash] = true
    data[hash] = code
    return true

boolean contains(x,y):
    code = code(x,y)
    hash = hash(x,y)
    while used[hash]:
        if data[hash] == code:
            return true
        else:
            hash = (hash + 1) mod SIZE
        return false

integer code(x,y): // convert pair x, y to a single value
    return (x << 32) or y

integer hash(x,y):
    r = (x << 16) or (y and 0xFFFF) mod SIZE //a simple hash function
    return r
```

Kết hợp danh sách liên kết với bảng băm

HashList kết hợp danh sách liên kết (tối ưu bộ nhớ và quá trình liệt kê lân cận của một đỉnh) và bảng băm (tối ưu quá trình tìm kiếm một cạnh).

Mã giả của việc biểu diễn đồ thị sử dụng HashList được mô tả như sau:

```
declare SIZE = 1000000, heads[], data[], used[], next[], used
procedure add(x,y):
    code = code(x,y)
```

```
        hash = hash(x,y)
        while used[hash]:
            if data[hash] == code:
                return false
            else:
                return hash = (hash+1) mod SIZE
        data[hash] = code
        used[hash] = true
        next[hash] = heads[x]
        heads[x] = hash
        return true

    boolean contains(x,y):
        code = code(x,y)
        hash = hash(x,y)
        while used[hash]:
            if data[hash] == code:
                return true
            else:
                hash = (hash + 1) mod SIZE
            return false

    procedure enumerate(x):
        for (i = heads[x] ; i <> -1; i = next) :
            y = data[i]
            // todo

    integer code(x,y): // convert pair x, y to a single value
        return (x << 32) or y

    integer hash(x,y):
        r = (x << 16) or (y and 0xFFFF) mod SIZE //a simple hash function
        return r
```

Ta đánh giá độ phức tạp về thời gian và bộ nhớ của các phương pháp như sau:

4.3.4 Giải thuật

Trong phần này, chúng tôi sẽ trình bày mã giả về giải thuật gom cụm của mô hình đề xuất.

Đầu vào: Đồ thị $G = (V, E)$

Đầu ra: Cụm các cộng đồng C sau khi được phát hiện.

Ý tưởng của thuật toán được trình bày dưới dạng mã giả như bên dưới.

Bảng 4.3. So sánh hai cách biểu diễn đồ thị, V là danh sách cạnh, E là danh sách đỉnh, K tập các cạnh nối đến với đỉnh u. Ta giả định rằng $|V| < |E|$.

	Danh sách liên kết	Bảng băm	HashList
Bộ nhớ lưu trữ	$O(\|V\|+\|E\|)$	$\boldsymbol{O(\|E\|)}$	$\boldsymbol{O(\|E\|)}$
Thêm một cạnh mới	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
Xóa một cạnh	$O(\|K\|)$	-	-
Tìm một cạnh	$O(\|K\|)$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$
Liệt kê các cạnh kề với một đỉnh u	$\boldsymbol{O(\|K\|)}$	$O(\|V\|)$	$\boldsymbol{O(\|K\|)}$

```
function getCluster(G=(V,E)):
    // loop
    while true:
        for i = 0 to V:
            C[i] <- {V[i]}
        // Phase 1
        while true:
            for u ∈ V and u ∈ c:
                ΔQ_u <- max ΔQ_u(u-> c2) ∀c2∈ C
                if ΔQ_u > 0 then:
                    remove(u,c) // remove u from c
                    add(u,c2) // add u to c2
            if no_vertex_moves_to_a_new_commmunity then:
                break while loop
        // Phase 2: Rebuild Graph
        V2 = C2
        E2 = calutate_new_weight(E,V2)
        V = V2
        E = E2
        if No community changes then:
            exit while loop
    return C

function newLouvain(G=(V,E), C):
    C = getCluster(G=(V,E))      // first, get community with louvain
    H = singleNodeCommunity(C)   // get single_node community as a subgraph
    C2 = getCluster(H)           // recluster with  H
    C_result = merge(C,C2)       // merge 2 community set
    return C_result
```

Ở đây, hàm *getCluster()* là hàm được xây dựng trên giải thuật Louvain với 2 giải đoạn:

Giai đoạn 1 (*Phase 1*) là duyệt tất cả các đỉnh và thực hiện việc gộp các đỉnh về 1 cộng đồng bằng tham lam nếu nó mang lại độ lợi về modularity cao hơn trước khi gộp.

Giai đoạn 2 (*Phase 2*) là xây dựng lại đồ thị mới với các đỉnh là các cụm mới sau khi được gộp ở giai đoạn 1 và trọng số đồ thị sẽ là tổng trọng số các cạnh liên kết từ 2 cụm được đại diện bởi hai đỉnh hiện tại.

Cả hai giai đoạn này được lặp đi, lặp lại cho đến khi cấu trúc cộng đồng của đồ thị không thay đổi nữa thì dừng lại.

Hàm *newLouvain()* mô tả bước cải tiến của chúng tôi về mặt giải thuật. Sau khi ta phát hiện được các cộng đồng bằng hàm *getCluster()*, ta thực hiện tìm và lấy ra cộng đồng chứa các *đỉnh đơn độc* bằng hàm *singleNodeCommunity()* và trả về kết quả ở dạng một đồ thị con. Sau đó, ta thực hiện phát hiện cộng đồng một lần nữa ở đồ thị con này. Kết quả sẽ được trộn (*merge*) với kết quả trước khi phát hiện và trả về kết quả cuối cùng. Số lần tìm tìm kiếm các cộng đồng chưa *đỉnh đơn độc* này có thể lặp đi lặp lại nhiều lần, riêng trong luận văn này, chúng tôi khuyến cáo chỉ cần thực hiện việc này 1 lần là đủ.

Chương 5

Hiện thực giải thuật

Trong chương này, chúng tôi sẽ trình bày chi tiết của việc hiện thực giải thuật được đề xuất. Đầu tiên, chúng tôi giới thiệu về ngôn ngữ lập trình cũng như các thư viện dùng để hiện thực trong toàn bộ hệ thống. Các bước sử dụng thư viện trong các giai đoạn. Cuối cùng, chúng tôi trình bày về hiện thực một ứng dụng cơ bản.

5.1 Ngôn ngữ lập trình và thư viện

Chúng tôi sử dụng *Python* và *C++* là hai ngôn ngữ lập trình chính cho toàn bộ hệ thống. Với khả năng tính toán mạnh mẻ và cộng đồng phát triển lớn, *Python* được xem là ngôn ngữ phổ biến nhất hiện nay trong khoa học dữ liệu. Python được sử dụng trong hầu hết các bước như thực hiện các bước tiền xử lý dữ liệu, tính toán, lưu trữ, hiển thị, kết quả. Đồng thời *C++* là ngôn ngữ có tốc độ tính toán và khả năng quả lý bộ nhớ tốt nhờ quản lý cấu trúc dữ liệu qua con trỏ (*pointer*). Trong luận văn, *C++* được sử dụng để hiện thực một phần cấu trúc dữ liệu và thuật toán gom cụm đồ thị *Louvain*.

Để hiện thực được mô hình gom cụm được đề xuất ở Chương 4, chúng tôi có sử dụng một vài thư viện để hỗ trợ quá trình hiện thực. Một số thư viện mà chúng tôi sử dụng trong luận văn là:

- **Networkx** là một thư viện mã nguồn mở trên ngôn ngữ lập trình *Python* cho phép tạo, sửa đổi, truy vấn, phân tích một mạng phức tạp. Networkx được tích hợp nhiều hàm và thủ tục xây dựng sẵn để phục vụ cho việc nghiên cứu, phân tích một mạng.

- **Matplotlib** là một thư viện vẽ sơ đồ 2D của *Python*, tạo ra các hình vẽ (*figure*) ở nhiều định dạng khác nhau và tương tác trên các nền tảng môi trường khác nhau. Matplotlib có thể được sử dụng trong các Python, các trình Python và IPython, Jupyter Notebook, các ứng dụng web và các bộ công cụ đồ họa với giao diện người dùng.

- **NumPy** là một thư viện cho ngôn ngữ lập trình *Python*, thêm hỗ trợ cho các mảng lớn, đa chiều và ma trận, cùng với một bộ sưu tập lớn các hàm toán học cấp cao để hoạt động trên các mảng này. Trong luận văn, NumPy được sử dụng để hỗ trợ quá trình tiền xử lý dữ liệu.

- **Python-louvain** là một thư viện của phát hiện cộng đồng dựa trên nền tảng Networkx. Thư viện hiện thực thuật toán phát hiện cộng đồng được mô tả ở bài báo *"Fast unfolding of communities in large networks"* [14].

5.2 Hiện thực giải thuật

Như đã trình bày ở Chương 4, chúng tôi thực hiện xây dựng lần lược các mô đun cho các giai đoạn chuẩn bị dữ liệu, bao gồm *Trích xuất đồ thị cục bộ* và *Trọng số hóa đồ thị* cũng như mô đun gom cụm dữ liệu, bao gồm *Gom cụm đồ thị* và *Hiển thị trực quan kết quả*. Dưới đây, chúng tôi sẽ trình bày tóm tắt việc hiện thực các mô đun trên:

Mô đun tiền xử lý dữ liệu nhận dữ liệu thô đầu vào, làm sạch, chuẩn hóa và lưu lại để phục vụ cho các mô đun tiếp theo. Dữ liệu về đồ thị ban đầu được lưu dưới dạng danh sách cạnh và chứa một số trường hợp *dị biệt* (*outlier*) như không có bạn bè nào hay có rất nhiều bạn (5000 bạn). Các người dùng này được cân nhắc để xóa khỏi dữ liệu. Nếu không có bạn nào hoặc có rất ít bạn (nhỏ hơn 5 bạn) thì ta xóa người dùng đó khỏi dữ liệu vì không có giá trị trong việc phân tích cộng đồng của những người dùng này. Một số người dùng khác có số bạn bè quá lớn được ta xem xét loại bỏ khỏi dữ liệu bằng cách xem số đỉnh bậc bằng 1 của họ nến lớn hơn 80%. Điều để phát hiện liệu họ có phải là người dùng bán hàng trực tuyến hay quảng cáo trên mạng hay không. Dữ liệu số lượng tin nhắn, thích và bình luận trên mạng, dữ liệu được lưu ở dạng *bản ghi lịch sử hoạt động* (*log*). Ta thực hiện việc đếm số lượng tin nhắn, bình luận và thích sau đó lưu lại dưới dạng từ điển có khóa là *mã định danh người dùng* (*uid*) và giá trị là một danh sách (*tuple*). Sau đó ta thực hiện loại bỏ một số trường hợp *dị biệt* vì số lượng tin nhắn quá lớn. Dữ liệu sau khi tiền xử lý xong cũng được nén lại dưới dạng danh sách kề đối với dữ liệu đồ thị và *JSON* [17] đối với dữ liệu tương tác.

Mô đun Trích xuất đồ thị cục bộ là một lớp (*class*) cho phép tải một đồ thị lớn (có thể là đồ thị toàn cục) lên và lưu dưới dạng *HashList* và cho phép gọi để lấy về một đồ thị cục bộ qua các hàm tiện ích (*utility function*). Đồ thị cục bộ được lấy bằng cách duyệt rộng (*breadth-first search*) từ đỉnh ứng với người dùng ta cần truy vấn 2 lần để lấy ra đồ thị con với các đỉnh là tập hợp của người dùng ta cần quan tâm và bạn bè của họ. Sau khi truy vấn được đồ thị, ta có thể chọn các xuất ra dưới dạng HashList, ma trận kề, danh sách kề hay danh sách cạnh tùy theo tùy chọn để phù hợp với bộ nhớ và thuận toán cần dùng.

Mô đun Trọng số hóa đồ thị phục vụ cho việc trọng số hóa đồ thị dựa vào các thông tin trên đồ thị và độ tương tác giữa các người dùng. Mô đun sẽ duyệt lần lượt từng cặp cạnh trên đồ thị và truy vấn số lượng tương tác từ cơ sở dữ liệu có sẵn sau đó áp dụng công thức (4.4) để tính trọng số của đồ thị. Kết quả, mô đun sẽ trả về một đồ thị mới là đồ thì đươc trọng số hóa từ đồ thị đầu vào của mô đun.

Mô đun Gom cụm đồ thị hiện thực giải thuật Louvain cải tiến như đã trình bày ở mục 4.3.2. Mô đun này nhận đầu vào là một đồ thị và trả về kết quả sau khi gom cụm là một từ điển (*Dictionary*) với các cặp khóa (*key*) là mã định danh (*id*) của một đỉnh và giá trị (*value*) là một số đại diện cho cộng đồng nó thuộc về.

Mô đun Hiển thị trực quan kết quả dùng để hiển thị trực quan lại đồ thị sau khi phát hiện cộng đồng xong. Sử dụng hàm *spring_layout* của thư viện Networkx để khởi tạo vị trí

các đỉnh của đồ thị, mô đun Hiển thị kết quả trực quan sẽ thiết lập màu của các đỉnh theo cộng đồng của nó. Sau đó ta có thể sử dụng thư viện Matplotlib để hiển thị dưới dạng ảnh, tập tin *pdf* hoặc các định dạng khác.

Các mô đun sử dụng các ngôn ngữ lập trình và các thư viện được mô tả ở Bảng 5.1 bên dưới:

Bảng 5.1. Các ngôn ngữ lập trình và các thư viện được sử dụng trong các mô đun.

Mô đun	**Ngôn ngữ lập trình**	**Thư viện**
Mô đun tiền xử lý dữ liệu	Python	Numpy
Mô đun Trích xuất đồ thị cục bộ	Python	Networkx
Mô đun Trọng số hóa đồ thị	Python	Numpy
Mô đun Gom cụm đồ thị	Python C++	Networkx Python-louvain
Mô đun Hiển thị trực quan kết quả	Python	Matplotlib Numpy Networkx

Chương 6

Thực nghiệm và đánh giá

Chúng tôi thực hiện thử nghiệm và đánh giá kết quả của việc phát hiện cộng đồng của các phương pháp được giới thiệu trong luận văn với bộ dữ liệu người dùng là một mạng xã hội phổ biến ở Việt Nam. Ta có sử dụng nhiều thang đo (*metric*) khác nhau để đánh giá kết quả dựa trên nhãn thực địa (*ground-truth*) của bộ dữ liệu.

6.1 Giới thiệu về bộ dữ liệu

6.1.1 Tổng quan về bộ dữ liệu

Bộ dữ liệu **X** mà chúng tôi đang xét là một đồ thị với 250000 đỉnh được trích tự một mạng xã hội tin nhắn tại Việt Nam. Dữ liệu trên được mã hóa dưới dạng mã định danh người dùng (*user identity*) và đưa vào một ít nhiễu (*noise*) nhỏ để đam bảo bảo mật thông tin cá nhân. Các nhãn cộng đồng thực địa từ dữ liệu trên được lấy từ các nhóm trò chuyện (*group chat*) và sử dụng là nhãn cho quá trình đánh giá. Một người dùng một lúc có nhiều nhóm chat cũng đại diện cho các cộng đồng khác nhau như *gia đình*, *bạn bè*, *đồng nghiệp*,... và danh sách các cộng đồng cũng được mã hóa về dạng mã định danh nhóm (*group indentity*). Sau khi xử lý và chọn lọc, ta trích xuất được 5000 đồ thị cục bộ đại diện cho những người dùng trên mạng xã hội này. Bảng 6.1 thống kê một số thông tin của 5000 đồ thị cục bộ này.

Bảng 6.1. Thống kê của các đồ thị cục bộ trích xuất từ tập dữ liệu **X** đang xét.

Thuộc tính	**Giá trị**
Số đỉnh trung bình	284.8
Số cạnh trung bình	1895.3
Số cộng đồng trung bình	9.3
Kích thước trung bình của cộng đồng	10.8

6.1.2 Tính chất đặc trưng của bộ dữ liệu

Bộ dữ liệu **X** có một tính đặc trưng khác với các bộ dữ liệu của các mạng xã hội khác như *Facebook*, *Twitter* là tính riêng tư (*privacy*) cao. Tính chất này có được do đặc thù của mạng xã hội nhắn tin sử dụng bạn bè trong danh bạ điện thoại làm nền tảng cho bạn bè của mạng xã hội.

Việc tập trung vào nhắn tin và thảo luận nhóm cũng như không công khai danh sách bạn về và danh sách những người xem bình luận và yêu thích tại các bài đăng của người dùng làm cho các mối quan hệ giữa hai người dùng thật sự chặc chẽ dù tồn tại ít hay nhiều bạn chung. Ngoài các nhóm lớn, các nhóm nhỏ từ 2 đến 3 người biết nhau cũng mang lại nhiều thông tin hữu dụng nhưng các mối quan hệ làm việc, trao đổi ngoài thực tế.

6.2 Phương pháp đánh giá

Sau khi các cộng đồng được phát hiện bằng các thuật toán phát hiện cộng đồng, nhiệm vụ tiếp theo của chúng ta là đánh giá độ tốt của cấu trúc cộng đồng vừa phát hiện. Việc đánh giá sẽ dễ dàng nếu ta có được các cấu trúc cộng đồng thực tế hay thực địa. Trong chương này, chúng tôi sẽ trình bày một vài độ đo của việc phát hiện cộng đồng dựa trên nhãn thực địa. Các độ đo chính là Precision, Recall, F-mesure và BER được tham khảo tại [18] và [19].

6.2.1 Độ đo Precision, Recall và F-measure

Ta sử dụng các độ đo sau để đánh giá kết quả của phương pháp so với dữ liệu có nhãn thực địa. Cho đồ thị $G = (V,E)$, gọi $C = C_1, C_2, ..., C_K$ là K cụm được gom cụm bằng các giải thuật phát hiện cộng đồng, $\hat{C} = \hat{C}_1, \hat{C}_2, ..., \hat{C}_L$ là L cộng đồng thực địa. Ta có độ đo sau

Precision là độ đo thể hiện độ chính xác trên các tập dự đoán.

$$Precision = \frac{1}{K} \sum_{k=1}^{K} \max_{j} \frac{|C_k, \hat{C}_j|}{|C_k|} \tag{6.1}$$

Recall là độ đo thể hiện độ nhạy của việc dự đoán.

$$Precision = \frac{1}{L} \sum_{l=1}^{L} \max_{j} \frac{|\hat{C}_l, C_j|}{|\hat{C}_l|} \tag{6.2}$$

F-measure được tính dựa vào công thức sau:

$$\text{F-measure} = \frac{2.Precision.Recall}{Precision + Recall} \tag{6.3}$$

6.2.2 Độ đo BER

Độ đo BER là độ đo để đo tỷ lệ lỗi cân bằng của kết quả dữ đoán và nhãn của dữ liệu. Đối với bài toán phát hiện công đồng, độ đo BER được tính như sau. Cho đồ thị $G = (V,E)$, gọi $C = C_1, C_2, ..., C_K$ là K cụm được gom cụm bằng các giải thuật phát hiện cộng đồng, $\hat{C} = \hat{C}_1, \hat{C}_2, ..., \hat{C}_L$ là L cộng đồng thực địa. Cho 2 cụm c và $\hat{c}$, ta có độ đo $BER(c, \hat{c})$ được tính bằng công thức:

$$BER(c, \hat{c}) = \frac{1}{2} \left(\frac{|c \setminus \hat{c}|}{|c|} + \frac{|\hat{c} \setminus c|}{|\hat{c}|} \right). \tag{6.4}$$

Ta sử dụng $RBER(c,\hat{c})$ được tính bằng $1-BER(c,\hat{c})$ và tìm được các tập $c \in C$ và $\hat{c} \in \hat{C}$ dựa trên phép gán tuyến tính sử dụng thuật toán Hungarian [20] để tìm giá trị tối đa của tổng các cặp theo giá trị $RBER$.

6.3 Kết quả

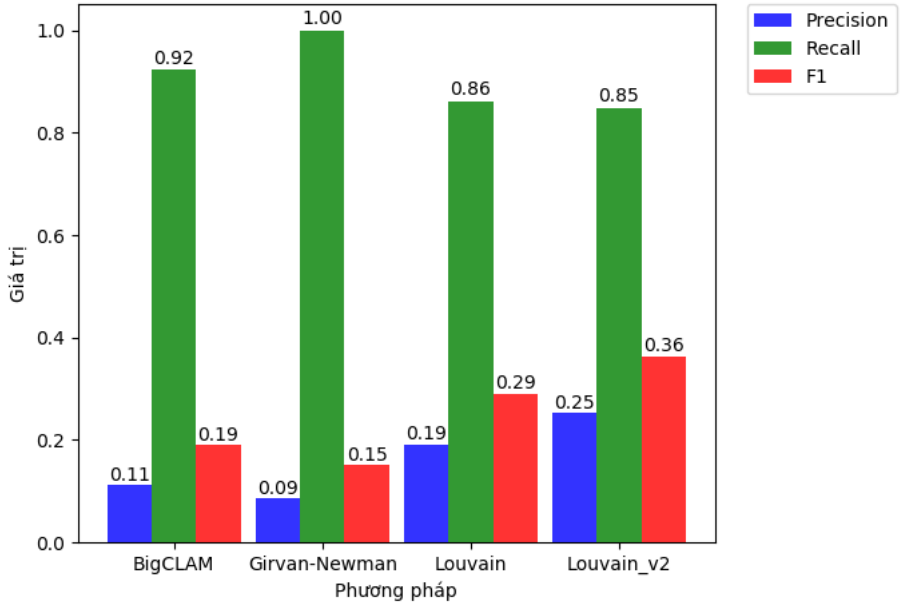
Sau khi sử dụng các phương pháp gom cụm được giới thiệu bao gồm giải thuật *Girvan-Newman*, *BigCLAM*, *Louvain* và so sánh với phương pháp *Louvain cải tiến* (*Louvain_v2*) trên tập dữ liệu **X** được trình bày ở Chương 6.1, ta thu được kết quả là cấu trúc các cụm. Trong mục này, chúng tôi sẽ trình bày kết quả của việc đánh giá các cụm sau khi được phát hiện.

6.3.1 Kết quả dựa trên độ đo Precision, Recall và F-measure

Sau khi sử dụng độ đo F-measure để đánh giá kết quả việc phát hiện cộng đồng, ta thu được kết quả tốt hơn so với phương pháp Louvain gốc [14], BigCLAM và Girvan-Newman. Kết quả chi tiết được trình bày ở Bảng 6.2.

Bảng 6.2. Kết quả đánh giá các phương pháp dựa vào Precision, Recall và F-measure

Phương pháp	**Precision**	**Recall**	**F-measure**
BigCLAM	0.11	0.92	0.19
Girvan-Newman	0.09	**1.00**	0.15
Louvain	0.19	0.86	0.29
Louvain v2	**0.25**	0.85	**0.36**

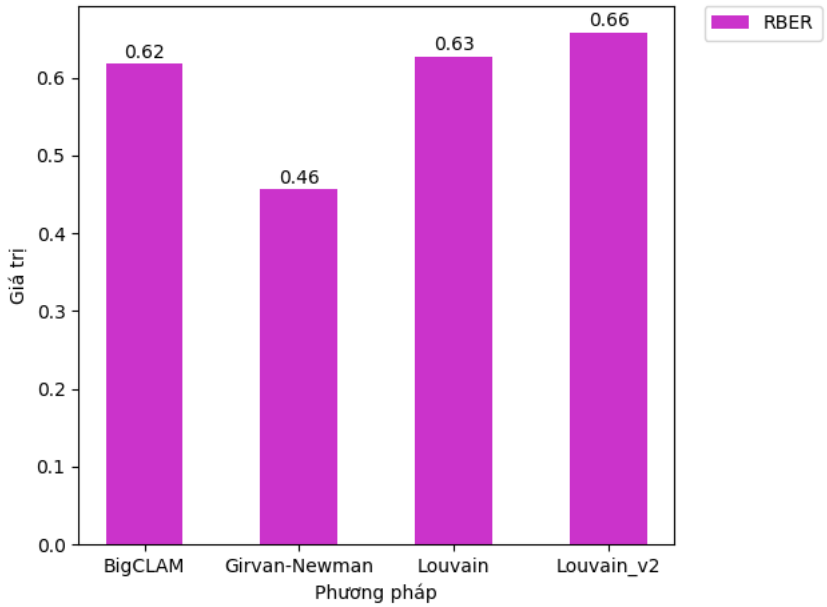


Hình 6.1. Biểu đồ so sánh kết quả của các phương pháp BigCLAM, Girvan-Newman, Louvain và Louvain cải tiến (louvain v2) sử dụng độ đo precision, recall và fmeasure tương ứng với Bảng 6.2

Đối với độ đo *BER*, kết quả ta thu được được chuyển về dạng *RBER*. Kết quả chi tiết được trình bày ở Bảng 6.3.

Bảng 6.3. Kết quả đánh giá các phương pháp dựa vào BER

Phương pháp	**RBER**
BigCLAM	0.62
Girvan-Newman	0.46
Louvain	0.63
Louvain v2	**0.66**

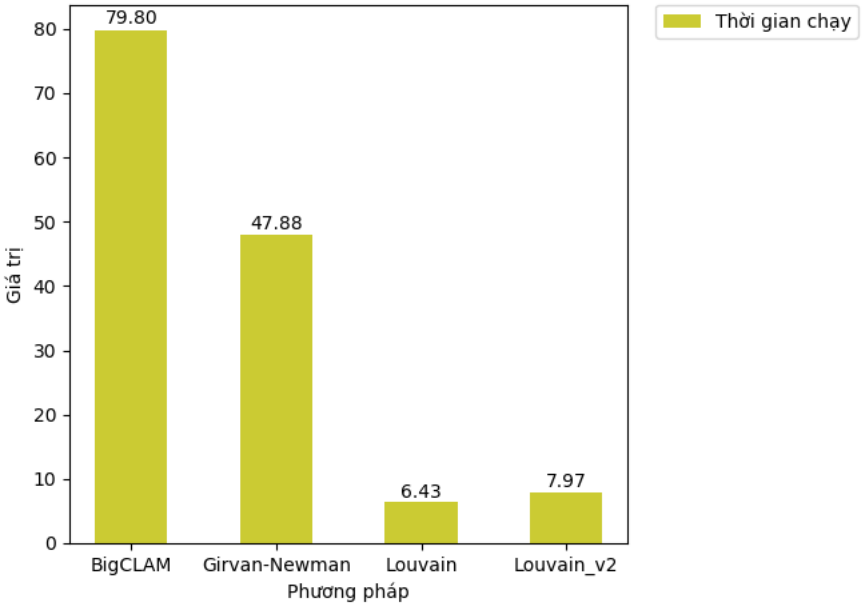


Hình 6.2. Biểu đồ so sánh kết quả của các phương pháp BigCLAM, Girvan-Newman, Louvain và Louvain cải tiến (louvain v2) sử dụng độ đo BER tương ứng với Bảng 6.3

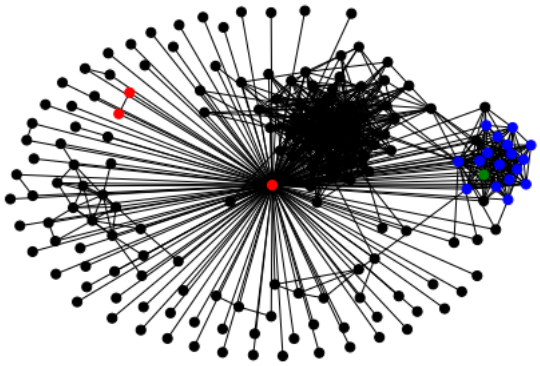
Ngoài ra, ta còn thực hiện so sánh về tốc độ thực thi của các phương pháp phát hiện cộng đồng.

Bảng 6.4. Thời gian thực thi các phương pháp phát phát hiện công đồng thực hiện chạy trên 100 đồ thị cục bộ

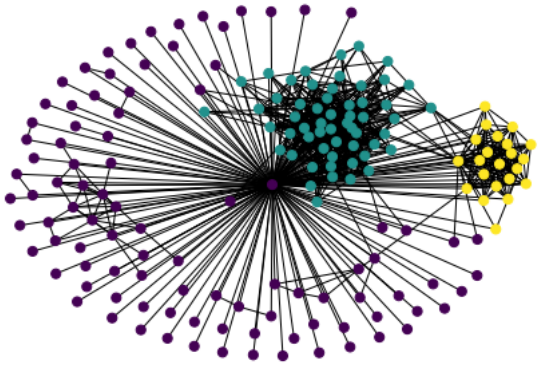
Phương pháp	**Thời gian thực thi**
BigCLAM	79.80
Girvan-Newman	47.88
Louvain	6.43
Louvain v2	7.96



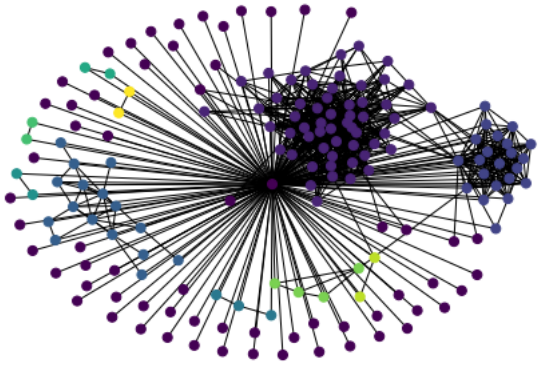
Hình 6.3. Biểu đồ thể hiện thời gian chạy của các phương pháp phát hiện cộng đồng thực hiện chạy trên 100 đồ thị cục bộ tương ứng với Bảng 6.4



(a) Nhãn thực địa của một đồ thị.

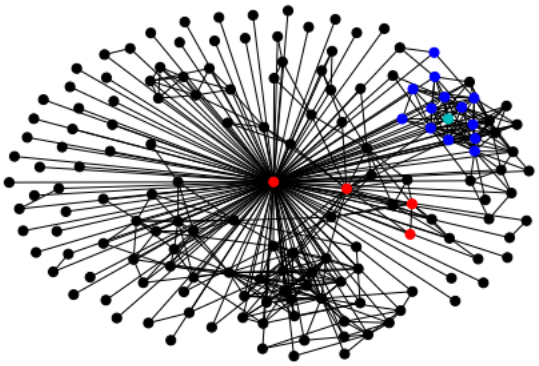


(b) Cộng đồng được dự phát hiện bằng phương pháp Louvain.

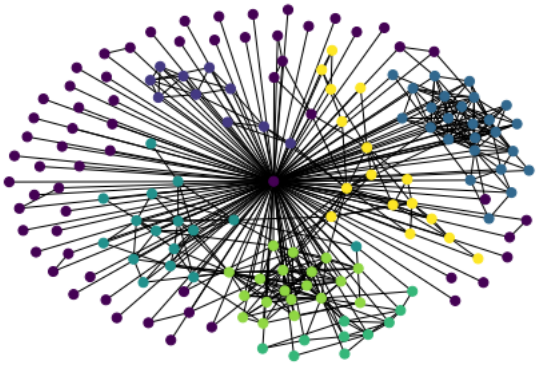


(c) Cộng đồng được phát hiện bằng phương pháp Louvain_v2.

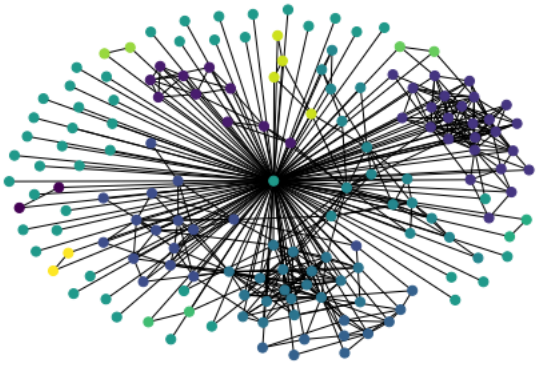
Hình 6.4. Một số kết quả về cộng đồng trên một đồ thị địa phương và nhãn thực địa.



(a) Nhãn thực địa của một đồ thị.



(b) Cộng đồng được dự phát hiện bằng phương pháp Louvain.



(c) Cộng đồng được phát hiện bằng phương pháp Louvain_v2.

Hình 6.5. Một số kết quả về cộng đồng trên một đồ thị địa phương và nhãn thực địa.

Chương 7

Tổng kết

7.1 Kết quả đạt được

Trong luận văn này, chúng tôi đã xây dựng được một mô hình phát hiện cộng đồng trên đồ thị và áp dụng mô hình để chạy trên một mạng xã hội thực tế. Mô hình dựa trên giải thuật Louvain [14] cải tiến lại giải thuật dựa vào các quan sát và kinh nghiệm cá nhân (*heuristics*). Đồng thời, chúng tôi có đề xuất thêm một độ đo dựa trên tương tác của người dùng mạng xã hội để trọng số hóa đồ thị. Chúng tôi sử dụng tập dữ liệu **X** của một mạng xã hội nhắn tin ở Việt Nam.

Kết quả đạt được dựa vào độ đo F-measure là 0.35 và dựa theo độ đo BER là 0.66. Tốt hơn phương pháp Louvain và các phương pháp khác. Ngoài ra, thời gian chạy của giả thuật Louvain cải tiến cũng nhanh hơn đáng kể so với các giải thuật khác cụ thể là thời gian chạy bằng 1.24 lần so với phương pháp Louvain trong khi đó phương pháp BigCLAM mất tới 12.41 lần và Girvan-Newman là 7.45 lần so với thời gian chạy của Louvain.

7.2 Hạn chế và hướng phát triển

Trong mô hình phát hiện cộng đồng đã hiện thực, chúng tôi chưa xử lý được các trường hợp trùng lặp trên cộng đồng. Ngoài ra, các cộng đồng được phát hiện chưa được định danh và khai thác ý nghĩa của nó.

Trong tương lai, chúng tôi sẽ cải tiếng thuật toán để phát hiện được các cộng đồng có trùng lặp cũng như khai phá thêm được ý nghĩa của các cộng đồng được phát hiện.

Tài liệu tham khảo

[1] J. Scott, *Social Network Analysis: A Handbook*. SAGE Publications Ltd, 2000.

[2] a. J. D. U. Anand Rajaraman, Jure Leskovec, *Mining of Massive Datasets*. Web Mining (mmds.org), 2014.

[3] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[4] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[5] J. Phipps, "Dendrogram topology," *Systematic zoology*, vol. 20, no. 3, pp. 306–308, 1971.

[6] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of Formalized Mathematics*, vol. 15, no. 9, pp. 237–247, 2003.

[7] J. Yang, "Overlapping community detection at scale: A nonnegative matrix factorization approach," *WSDM '13*, pp. 587–596, 2013.

[8] J. Yang and J. Leskovec., "Community-affiliation graph model for overlapping network community detection," *ICDM '12*, 2012.

[9] J. Yang and J. Leskovec., "Structure and overlaps of communities in networks," *SNAKDD '12*, 2012.

[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[11] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Physica-Verlag HD*, 2010.

[12] F. O. de Franca and G. P. Coelho, "A flexible fitness function for community detection in complex networks," *arXiv preprint arXiv:1406.2545*, 2014.

[13] F. O. de França and G. P. Coelho, "A flexible fitness function for community detection in complex networks," in *Complex Networks VI*, pp. 1–12, Springer, 2015.

[14] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[15] A. Arenas, J. Duch, A. Fernández, and S. Gómez, "Size reduction of complex networks preserving modularity," *New Journal of Physics*, vol. 9, no. 6, p. 176, 2007.

[16] M. A. Kolosovskiy, "Data structure for representing a graph: combination of linked list and hash table," *arXiv preprint arXiv:0908.3089*, 2009.

[17] T. Bray, "The javascript object notation (json) data interchange format," 2017.

[18] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, pp. 539–547, 2012.

[19] J. Artiles, J. Gonzalo, and S. Sekine, "The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task," in *Proceedings of the 4th International Workshop on Semantic Evaluations*, pp. 64–69, Association for Computational Linguistics, 2007.

[20] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.